TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Informatik Institut für Künstliche Intelligenz · Professur für Bildverarbeitung

COMPARISON OF LEARNED INFERENCE APPROACHES FOR IMAGE RESTORATION

Master-Arbeit
zur Erlangung des Hochschulgrades

Master of Science

(M. Sc.)

vorgelegt von: JAKOB KRUSE
geboren am: 21. AUGUST 1990 in DRESDEN

Tag der Einreichung: 12. SEPTEMBER 2016
Betreuer: Prof. Ph.D. Carsten ROTHER

Abstract

A common approach to many image restoration problems is to model the corruption process and the desired characteristics of restored images with the help of a random field. Then Bayesian inference can be carried out to find the best estimate of the original image for any given observation. In several recent works, the inference procedure is truncated to a small number of iterations to guarantee fast runtime. The loss in accuracy is compensated for by discriminatively learning an individual set of model parameters for each iteration, minimizing a loss function over training data. The resulting application-specific architectures can achieve high restoration quality with small computational effort.

For this thesis, two truncated inference approaches were implemented and trained for image denoising and non-blind deblurring. The gradient-descent approach, which had previously been reported to produce very good denoising results, is shown here to be less suitable for the deblurring task, except when truncating inference after many iterations or supplying strong initial guesses. A variant of half-quadratic inference, which has not been used in this context before, produces good deblurring results within few iterations, at the cost of a much longer computation time. Both approaches are shown to be equally suited for the denoising task, although each works best for a different type of image content.

Zusammenfassung

Ein beliebter Ansatz zum Lösen von Problemen der Bildwiederherstellung besteht darin, den Korruptionsprozess und die gewünschten Eigenschaften wiederhergestellter Bilder mit Hilfe eines Random Field-Modells zu beschreiben. Darauf lässt sich Bayes'sche Inferenz anwenden, um zu einer gegebenen Beobachtung die beste Vermutung für das Ursprungsbild zu finden. In einigen neueren Werken wird diese Inferenz auf eine kleine Zahl an Iterationen begrenzt, um schnelle Laufzeit sicherzustellen. Der Verlust an Genauigkeit wird ausgeglichen, indem für jede Iteration ein separater Satz an Modellparametern diskriminativ von Trainingsdaten gelernt wird. Die so entstandenen anwendungsspezifischen Systeme können mit geringem Rechenaufwand eine hohe Wiederherstellungsqualität erzielen.

Im Zuge dieser Arbeit wurden zwei derart begrenzte Inferenzverfahren umgesetzt und jeweils zum Entfernen von Rauschen und Unschärfe trainiert. Das Gradientenverfahren, dessen starke Ergebnisse beim Entrauschen bereits publiziert wurden, zeigt sich hier als weniger geeignet zum Entfernen von Unschärfe, wenn die Inferenz nicht erst deutlich später begrenzt wird oder gute Ausgangsvermutungen bereitgestellt werden. Eine Variante halb-quadratischer Inferenz, die noch nicht in diesem Kontext angewendet wurde, führt dagegen in wenigen Iterationen zu guten Ergebnissen, benötigt jedoch erheblich längere Rechenzeit. Beide Ansätze sind gleichauf beim Entfernen von Bildrauschen, erreichen ihre jeweils besten Ergebnisse aber bei unterschiedlichen Bildinhalten.

Acknowledgements

I wish to thank Uwe Schmidt for his invaluable input on this thesis, from explaining high-level concepts to providing detailed criticism and feedback. While working on this topic, I learned many things I might never have discovered on my own. I also want to thank my parents, who, among many wonderful things, made it possible for me to pursue the educational path that led to this point. Finally, my university life could never have been as fulfilling without Lisa, who was a great source of support during the sometimes quite stressful past months. Thank you :)

Contents

1 Introduction **9**
- 1.1 Related Work . . . 10
- 1.2 Outline of Thesis . . . 11

2 Background and Theory **13**
- 2.1 Notation . . . 13
- 2.2 Images and Restoration . . . 15
  - 2.2.1 Image Representation . . . 16
  - 2.2.2 Noisy Images . . . 17
  - 2.2.3 Blurred Images . . . 18
  - 2.2.4 Finding the Restored Image . . . 20
- 2.3 Graphical Models . . . 20
  - 2.3.1 Markov Random Fields . . . 21
  - 2.3.2 Fields of Experts . . . 24
  - 2.3.3 Robust Potentials . . . 25
  - 2.3.4 Maximum A-Posteriori Inference . . . 26
  - 2.3.5 Energy Minimization . . . 28

3 Learned Inference **31**
- 3.1 Bi-level Optimization . . . 31
  - 3.1.1 Loss Function . . . 32
  - 3.1.2 Training Data . . . 34
  - 3.1.3 Model Parameters . . . 35
- 3.2 Truncated Optimization . . . 39
  - 3.2.1 Computing Gradients with Backpropagation . . . 41
  - 3.2.2 Greedy and Joint Training . . . 42
- 3.3 Inference via Gradient Descent . . . 43
  - 3.3.1 Motivation . . . 43
  - 3.3.2 Energy Gradient . . . 43
  - 3.3.3 Update Formula . . . 44
  - 3.3.4 Derivative of the Loss Function . . . 45
  - 3.3.5 Input Gradient . . . 45
  - 3.3.6 Parameter Gradients . . . 46
  - 3.3.7 Summary . . . 48
- 3.4 Half-Quadratic Inference . . . 49
  - 3.4.1 Motivation . . . 49
  - 3.4.2 Update Formula . . . 51
  - 3.4.3 Solving Systems of Linear Equations . . . 52
  - 3.4.4 Derivative of the Loss Function . . . 54
  - 3.4.5 Input Gradient . . . 55
  - 3.4.6 Parameter Gradients . . . 56

3.4.7 Summary 59

4 Experiments and Results **61**
4.1 Training and Test Setup 61
4.1.1 Data Sets 62
4.1.2 Boundary Handling 63
4.1.3 Initialization 64
4.2 Image Denoising 66
4.2.1 Gradient Descent 68
4.2.2 Half-Quadratic Inference 69
4.3 Image Deblurring 70
4.3.1 Gradient Descent 71
4.3.2 Half-Quadratic Inference 72

5 Discussion and Outlook **77**
5.1 Denoising 77
5.2 Deblurring 78
5.3 Computation time 82
5.4 Outlook 83
5.5 Conclusion 84

6 Appendix **87**

Bibliography **95**

1 Introduction

The field of *computer vision* is a branch of computer science dedicated to the extraction of information from visual media like photographs, microscope or telescope output, stereo images and video streams. The information might be a segmentation of an image into objects and background, an estimation of an object's movement or a reconstruction of the three-dimensional structure of the scene. For these tasks, computer vision uses techniques from the related fields of signal processing, pattern recognition and machine learning among others.

One of the classical problems in computer vision is that of *image restoration*. Having observed some corrupted image, the restoration task is to produce the best possible estimate of the original, uncorrupted image. However, like many problems in computer vision, the image restoration problem is usually not well-posed because the input images lack important information. For an algorithm to make decisions when faced with such ambiguity, some form of prior knowledge about the desired properties of the output images is necessary.

A common technique for encoding this kind of information is to model an image as a graph structure, where each node corresponds to one image pixel and the edges represent their neighbourhood relationships. An *energy* function can then be defined on the whole graph which assigns a score to each possible configuration of the pixel values. Less desirable configurations are assigned a higher energy, which translates to a lower probability under the model. With this resulting *prior* probability distribution over all images, as well as a model for the *likelihood* of any given clean image leading to the observed corrupted image, Bayesian inference can be employed to reason about the unknown quantity, which is the original image.

The task of finding the candidate for the restored image with the highest *posterior* probability under a specified model is called *maximum a-posteriori inference*, and is commonly carried out by *minimizing* the energy function. This can be achieved using a wide range of known gradient-based optimization techniques. One disadvantage of this approach, however, is that minimization of the energy can take many iterations to converge. Furthermore, it is not usually clear in advance how long the inference procedure will take.

An alternative approach proposed in recent years, sometimes labelled *truncated optimization*, aims to avoid this. The central idea is to limit the minimization procedure to a fixed small number of steps, but to train the model parameters separately for each step with respect to a loss function over training data. Essentially, each step uses a differently trained model, tuned to improve on the result of its predecessor. The whole arrangement can be interpreted as a *cascade*. As a result, the model architecture and the chosen gradient-based optimization algorithm are no longer independent, but become fused in one unit.

While many such combinations of model and inference algorithm are possible, only some have been studied in the literature so far, and there has been no systematic comparison as of yet. While it is beyond the scope of this work to provide an exhaustive survey, the following chapters will take a close look at two combinations and see how they compare to each other when applied to specific image restoration problems.

Both will be based on the *Field of Experts* model architecture [35], a higher-order graphical model that is very popular as a prior for natural images. The first approach combines this with the standard *gradient descent* approach for energy minimization, a combination that has been studied before in [13]. The second inference approach will be based on a variant of *half-quadratic inference* [19], which to the best of our knowledge has not yet been studied in combination with this type of model.

Finally, the concrete task for this thesis was to implement these two model-inference combinations from scratch and train them under identical conditions. Then a direct comparison of both the quality of the results after each inference step and the required runtime can be made. For this comparison, the two classical image restoration problems of *denoising* and *non-blind deblurring*, both under the assumption of Gaussian noise, are considered here.

The application of the *gradient descent* approach to the deblurring problem, as well as the application of the considered *half-quadratic* variant in such a truncated optimization setting in general, constitute new contributions to the field.

1.1 Related Work

The *Field of Experts* prior was proposed by Roth and Black in [34] as a higher-order variant of the common *Markov random field* architecture, particularly suited for image restoration. It models the probability of an image as the product of potentials of overlapping image patches called *cliques*. Each potential consists of a nonlinear expert function modelling the response of a linear filter. Of special interest is the ability to learn all model parameters from training data, which will be exploited to train the models in this thesis. The authors also already advocate the simple gradient descent algorithm as a means of finding an approximate maximum a-posteriori solution.

Where Roth and Black chose a generative training approach, employing *sampling* and *contrastive divergence*, Samuel and Tappen [39] suggest training the *Field of Experts* model discriminatively with respect to a specific loss function. Instead of training the model independently on natural images and doing inference later on, they argue that the model parameters should be tuned to maximize the effectiveness of a particular inference method to achieve the best performance. This leads to a *bi-level optimization problem*: In the lower level, the energy of the random field configuration is minimized to find a maximum a-posteriori estimate for a given input image. In the upper level, the model parameters are optimized such that the results from the lower level are as close as possible to ground truth data, according to a chosen measure of similarity.

The notion to *truncate the optimization* in the lower level, that is to limit the inference procedure to a fixed number of steps, can first be found in Barbu's work [2]. There,

an *Active Random Field* is defined as the combination of a random field model and an inference algorithm. The author chooses the *Field of Experts* architecture for the image model. The same bi-level optimization as in [39] is applied, except that the lower level consists of only 1 to 4 iterations of standard *gradient descent* to approximate an energy minimum. It is shown that the suboptimal solution obtained in the lower level is compensated for by the model bias introduced through loss-specific training. The truncated optimization approach yields results on par with the original *Field of Experts* paper [34], while achieving a speedup of several orders of magnitude. A closely related route is taken by Domke in [17]. Both authors train a set of arbitrary linear filters, shared between all steps of their models, and the parameters for a very constrained set of expert functions.

In a similar way, cascades of *regression tree fields* [26], a novel form of Gaussian random fields, are discriminatively trained by Schmidt *et al.* in [41] for the tasks of image denoising and deblurring. In their case, both the nonlinear functions and the linear filters are highly flexible and trained individually for each stage of the cascade, an idea that had not been exploited in previous truncated optimization schemes.

In [14], Chen *et al.* revisit the discriminative approach from [39] and show that, using an improved training procedure, they can achieve much more competitive results than previously assumed. Their findings suggest that solving the lower level problem with higher accuracy leads to better overall performance of the bi-level framework. They also note that the maximum a-posteriori estimate remains a very effective inference option for low-level computer vision tasks such as denoising.

Building on [41], *Cascades of Shrinkage Fields* are introduced by Schmidt and Roth in [40] as a combination of the *Field of Experts* model with an efficient variant of the *additive form* of *half-quadratic inference* [20]. They again utilize a cascade of fixed length for the lower level of optimization, where each stage represents one iteration of half-quadratic optimization.

The idea of training each step of a truncated model individually is transferred back to the *Active Random Field* approach by Chen and Pock in [13]. As in Barbu's work [2], a *Field of Experts* architecture is paired with truncated *gradient descent* inference. However, the new approach differs in that the gradient descent steps no longer share their parameters. The authors then find that, in contrast to common random field applications, their freely tuned expert functions work not just to locally ensure smoothness of the image, but also to sharpen edges and even encode preferences for certain textures or patterns. For the task of image denoising, their best model outperforms all competitors while requiring significantly shorter computation time.

1.2 Outline of Thesis

This work is divided into five parts. Following this introduction, Chapter 2 covers a number of basic assumptions and definitions regarding the images on which the examined algorithms operate. A brief overview of the processes behind noise and blur in natural images is provided, followed by a discussion of the image restoration problem in general. The chapter also gives an introduction to Markov random fields and related concepts, which are essential for understanding the model architectures that are be employed later on.

Chapter 3 gives the mathematical formulation of the bi-level optimization problem, shows how it is modified by truncating the lower level, and covers the two approaches used here for finding an approximate solution. Starting with the concept of loss-specific training, the chapter covers the loss function and the required training data as well as the parameterization of all learned model components, most notably the linear filters and the nonlinear functions. After that, the idea of truncating the inference procedure is presented, and what this means in terms of the model formulated before. As will be shown, the truncation allows the application of a back-propagation scheme to efficiently calculate the gradients of the model parameters for each inference step with respect to the value of the loss function. After that follows a short discussion of the differences between joint and greedy training of said parameters, before the two concrete inference approaches are presented in detail. For each of these, there is a brief section on motivation followed by the resulting update formula for one inference iteration. Next, derivations of the parameter gradients are presented for each approach, including some remarks on how they can be computed more efficiently.

After that, Chapter 4 is dedicated to the practical evaluation of the two truncated optimization frameworks. After a number of general and technical points regarding the experimental setup, the results of different variants of the two model-inference combinations for the tasks of image denoising and deblurring are presented. The performance of the trained models on established test sets is shown next to the results of related work, and the computation time of the new implementation reported for each variant.

Finally, Chapter 5 is used to draw conclusions from the experimental results, compare them to each other and to initial expectations, and offer explanations for the observed differences. It discusses the weak and strong points of each approach and addresses the matter of computation time. In addition, it provides an overview of possible extensions and improvements of the work presented in this thesis, as well as ideas for further research on the topic.

2 Background and Theory

To enable a good understanding of both the task that needs to be solved and the underlying mathematical models, this chapter will introduce a number of relevant concepts from the field of *computer vision*. After a short section on notation and a few basic mathematical results, some fundamental definitions and assumptions are stated about the images that are processed by the models in this thesis. This includes how the images are represented, as well as the basic models for image degradation by noise and by blur. From there the chapter will explain how these ideas connect to the random field models that are the basis for the approaches compared in this thesis, and see how they can help with restoring corrupted images.

2.1 Notation

Throughout this work there will be a few rules regarding notation that should make it easier to discern, at a glance, what each of the symbols in an equation is supposed to represent.

A lower case letter in bold, like $\mathbf{x}$, will always represent a *column vector*. Its transpose is $\mathbf{x}^\top$. An upper case letter in bold, like $\mathbf{F}$, will represent a *matrix*. Its transpose is $\mathbf{F}^\top$, its inverse $\mathbf{F}^{-1}$. $\mathbf{I}^{k\times k}$ is the identity matrix of size $k \times k$. Anything not set in bold will represent either a *scalar*, like λ, or a *function*, like $\rho(x)$.

The term $\operatorname{diag}\{\mathbf{x}\}$ represents a *diagonal matrix* with height and width equal to the dimension of the vector $\mathbf{x}$, where the diagonal entries are given by the elements of $\mathbf{x}$, and all other entries are zero.

Depending on context, an index, as in $\mathbf{x}_t$, will mark a specific *member out of a group* of similar objects or, as in x_n, the n^{th} *element of a structure*. If both kinds of indices are needed, the latter will appear second, separated by a comma, as in $x_{t,n}$. For a matrix $\mathbf{A}$, the entry in the i^{th} row and j^{th} column is denoted A_{ij}. Similarly, for a vector $\mathbf{a}$, the i^{th} element of $\mathbf{a}$ is a_i.

A hat, like in $\hat{\mathbf{x}}$, means that the variable in question is an *estimate* of another, *unknown quantity*, which will sometimes be marked with an asterisk, as in $\mathbf{x}^*$.

The *convolution* of a vector $\mathbf{x}$ with a second vector $\mathbf{k}$ will be written as $\mathbf{x} * \mathbf{k} = \mathbf{k} * \mathbf{x}$.

The *Gaussian normal distribution* with the *variance* σ^2 and centered on the *mean* μ, which is defined as

$$\mathcal{N}(\mu, \sigma^2) \coloneqq \frac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

will sometimes be denoted $\mathcal{N}(x; \mu, \sigma^2)$ to stress that it describes the distribution of some variable x. Analoguous notation is used for the *multivariate* Gaussian normal

distribution $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of a vector $\mathbf{x}$, centered on the vector $\boldsymbol{\mu}$ and defined via the *covariance matrix* $\boldsymbol{\Sigma}$. This is also called the *moment parameterization*. When the distribution is given in *information parameterization* [44] instead, it is written as $\mathcal{N}_I(\mathbf{x}; \boldsymbol{\eta}, \boldsymbol{\Omega})$ with the *information vector* $\boldsymbol{\eta} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ and the *precision matrix* $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$.

When it comes to *matrix calculus*, that is, differentiating anything that involves non-scalar expressions, this work will always follow the *numerator layout* convention[1], also known as the *Jacobian formulation*, for consistency. Thus differentiating a vector $\mathbf{y} \in \mathbb{R}^m$ with respect to a vector $\mathbf{x} \in \mathbb{R}^n$ will yield an $m \times n$ matrix of the partial derivatives. Following the *denominator layout* instead would lead to the same results, only transposed.

For some of the derivations in Chapter 3 it will be convenient to apply binary operators to vectors or matrices separately for each element. The *element-wise* matrix product is usually known as the *Hadamard* or *Schur product* [32] and will be written as $\mathbf{A} \odot \mathbf{B}$ with $(\mathbf{A} \odot \mathbf{B})_{ij} = A_{ij} \cdot B_{ij}$.

The principle can be illustrated with this simple example:

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \odot \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 10 \\ 6 & 6 \end{bmatrix}$$

In the case of vectors, $(\mathbf{a} \odot \mathbf{b})_i = a_i \cdot b_i = (\mathrm{diag}\{\mathbf{a}\} \cdot \mathbf{b})_i$. A simple example of this would be the following:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \odot \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \\ 18 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

If an element-wise binary operator appears between a vector and a matrix, it shall be applied to each column of the matrix separately, setting

$$(\mathbf{a} \odot \mathbf{B})_{ij} = (\mathbf{B} \odot \mathbf{a})_{ij} \coloneqq a_i \cdot B_{ij}\,. \tag{2.1}$$

This is an extension of the conventional definition, but it will come in very handy later on. For example, it allows the formulation $(\mathbf{a} \odot \mathbf{B})_{ij} = a_i \cdot B_{ij} = (\mathrm{diag}\{\mathbf{a}\} \cdot \mathbf{B})_{ij}$ for a vector-matrix pair as well:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \odot \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} = \begin{bmatrix} 4 & 7 \\ 10 & 16 \\ 18 & 27 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \cdot \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}$$

It also allows for the introduction of an *element-wise product rule* for two vectors $\mathbf{a}$

[1] see for example `https://en.wikipedia.org/wiki/Matrix_calculus`

and $\mathbf{b}$, where both depend on a third vector $\mathbf{x}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{x}}(\mathbf{a} \odot \mathbf{b}) &= \begin{bmatrix} \vdots \\ \frac{\partial}{\partial \mathbf{x}}(a_i \cdot b_i) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ a_i \frac{\partial b_i}{\partial \mathbf{x}} + b_i \frac{\partial a_i}{\partial \mathbf{x}} \\ \vdots \end{bmatrix} \\ &= \begin{bmatrix} \vdots \\ a_i \frac{\partial b_i}{\partial \mathbf{x}} \\ \vdots \end{bmatrix} + \begin{bmatrix} \vdots \\ b_i \frac{\partial a_i}{\partial \mathbf{x}} \\ \vdots \end{bmatrix} = \mathbf{a} \odot \frac{\partial \mathbf{b}}{\partial \mathbf{x}} + \mathbf{b} \odot \frac{\partial \mathbf{a}}{\partial \mathbf{x}} \end{aligned} \tag{2.2}$$

Note that both $\frac{\partial \mathbf{b}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{a}}{\partial \mathbf{x}}$ are matrices.

Another result that will be of use later on is the following equality for two vectors $\mathbf{a} \in \mathbb{R}^p, \mathbf{c} \in \mathbb{R}^q$ and a matrix $\mathbf{B} \in \mathbb{R}^{p \times q}$:

$$\begin{aligned}((\mathbf{a} \odot \mathbf{B}) \cdot \mathbf{c})_i &= \sum_{j=1}^{q} (a_i \cdot B_{ij}) \cdot c_j \\ &= a_i \cdot \sum_{j=1}^{q} B_{ij} \cdot c_j \\ &= a_i \cdot (\mathbf{B} \cdot \mathbf{c})_i \\ &= (\mathbf{a} \odot (\mathbf{B} \cdot \mathbf{c}))_i \,,\end{aligned}$$

and therefore

$$(\mathbf{a} \odot \mathbf{B}) \cdot \mathbf{c} = \mathbf{a} \odot (\mathbf{B} \cdot \mathbf{c}) \,. \tag{2.3}$$

While many further results could be found for element-wise operators, these are the ones immediately relevant to this thesis.

2.2 Images and Restoration

The task of *image restoration* is one of the classical problems in low-level computer vision[2]. Generally speaking, it is the process of estimating the true original image from a corrupted observation or a set thereof.

Such corruptions of the observed image can have a multitude of forms and causes. While this thesis will only deal with *noise* and *blur*, arguably the best known sources of corruption, other sources have been and continue to be the subject of research.

Many operations commonly performed on images in everyday use, like *downscaling* and *lossy compression*, degrade the image in typical ways. These can be addressed by techniques like *super-resolution* [45] or *JPEG deblocking* [26, 13]. Problems during *data transmission*, as well as undesired *overlays*, such as text pasted over the image, can lead to whole image regions being lost. A common countermeasure in these

[2] *Low-level* computer vision tends to deal with local, pixel-based tasks as opposed to *high-level* problems such as face recognition and scene understanding. There is, however, no clear-cut definition of what constitutes high-level and low-level computer vision.

situations is *inpainting* [22], where the unknown parts are interpolated based on the surviving observation and prior knowledge. Standard digital cameras also often introduce *colour aberrations* as a result of the way they capture and interpolate colour samples in an internal step called *demosaicing* [30].

2.2.1 Image Representation

Before having a closer look at the models for image corruption used in this work, there should be a few words on what is meant by the word *image* in this context.

First of all, the focus of this work is on *natural images*, which generally means photographs of real-world scenes captured by digital cameras. The models and techniques presented later on certainly extend to other types of images, such as ultrasound scans or digital renderings, but these will not be investigated here.

A significant part of the scientific community generally conceptualizes images as *two-dimensional signals* in the spatial domain, continuous functions that have to be sampled from to obtain concrete values, for example at pixel positions. This fits well with the physical world the images come from, where structures and surfaces above the scale of individual particles tend to be smooth and continuous. It also allows for the full range of signal processing techniques, shared with other areas of research, to be applied [42].

For the approaches discussed here, however, this understanding of images is not particularly useful. Instead, an image will be treated as just a two-dimensional *matrix of variables*, each representing the intensity of one specific pixel. For coloured images, there is one such matrix for each colour channel. These can be combined into a single three-dimensional array. For the sake of simplicity, this thesis will only consider grayscale images, but it is worth noting that the methods used can be generalized to handle colour images as well.

Although the images produced by a normal camera are made up of discrete integers, with values usually ranging from 0 to 255, the variables of the image model used here are defined as continuous real numbers. This allows for greater accuracy, but more importantly it guarantees that the equations presented later on are differentiable, which will be important for parameter learning. The restored images resulting from the algorithms discussed later on can always be converted back to integer values to comply with common file formats.

Having introduced this matrix representation of images, it is often helpful to move to a different representation which is more convenient from a mathematical perspective. An image matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ can be reshaped into a vector $\mathrm{vec}(\mathbf{M}) = \mathbf{v} \in \mathbb{R}^{mn}$ simply by concatenating all the columns of $\mathbf{M}$, as illustrated in Figure 2.1. Note that this can easily be reverted. For the remainder of this thesis, the vector representation will be used when images occur in equations, unless explicitly stated otherwise.

Furthermore, when talking about images in the context of image restoration there are some terms that may be used interchangeably. The *clean image*, which is unknown to the restoration algorithm, is also called *original image* or *ground truth* and will always be denoted by $\mathbf{x}_{\mathrm{gt}}$. The *corrupted image* based on it, which is the *input image* for the algorithm, is also called the *observation* and will always be denoted by $\mathbf{y}$.

$$\begin{bmatrix} 1.0 & 5.0 & 9.0 & 13.0 \\ 2.0 & 6.0 & 10.0 & 14.0 \\ 3.0 & 7.0 & 11.0 & 15.0 \\ 4.0 & 8.0 & 12.0 & 16.0 \end{bmatrix} \Longleftrightarrow \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ \vdots \\ 14.0 \\ 15.0 \\ 16.0 \end{bmatrix}$$

Figure 2.1: Illustration of reversible matrix-vector conversion

Finally, the *output image* of the restoration algorithm is usually called *prediction* and will always be denoted $\hat{\mathbf{x}}$, since it can only ever be an estimate of the original image.

This assignment of $\mathbf{y}$ and $\hat{\mathbf{x}}$ as input and output may be confusing to some readers, as these variables are often used the other way around. A potential reason why it is different in the context of image restoration is that the problem is an *inverse* one: $\mathbf{y}$ is in fact the output of some image corruption process with input $\mathbf{x}$. But for an algorithm seeking to undo the corruption, it is the input. Switching the names around here would likely only cause additional confusion.

2.2.2 Noisy Images

Maybe the most obvious and well-understood type of image corruption is noise. Just like image corruption in general, noise in natural images can stem from a variety of very different sources.

There is the so-called *shot noise*, which arises from the inherent randomness when photons from a light source reach the sensor. In theory, *shot noise* is governed by a Poisson distribution, but under most conditions it can be reasonably described by a normal distribution about its mean. It is more prevalent in darker image regions, where the total number of photons collected is lower.

The same *demosaicing* process responsible for the afore-mentioned colour aberrations in conventional digital cameras also introduces noise [12], as do other electronic circuits that process the captured data. Temperature plays an important role in this respect, as heated sensors are more prone to leak noise into the image [15, § 11.2.3].

Finally, the true image intensities will always be rounded to a value that is representable in the chosen image format, often an 8 bit integer in the range $[0, 255]$, in a step called *quantization*. This is, in essence, just another form of noise, although this time following a uniform distribution.

However, modelling each type of noise individually and correctly is not in the scope of this work. To simplify matters, a common assumption for image noise is a single Gaussian distribution, affecting each pixel x_i of an image $\mathbf{x}$ in the following way:

$$y_i = x_i + r, \qquad r \sim \mathcal{N}(0, \sigma^2) \tag{2.4}$$

Figure 2.2 shows the effect of such Gaussian white noise applied to an image from the data set used for the denoising experiments in Chapter 4.

(a) $\sigma^2 = 0$ **(b)** $\sigma^2 = 5$ **(c)** $\sigma^2 = 15$ **(d)** $\sigma^2 = 25$ **(e)** $\sigma^2 = 50$

Figure 2.2: Different levels of Gaussian white noise applied to a natural image taken from the *Berkeley Segmentation Data Set* [1]. The noise level σ^2 determines how heavily the image is corrupted, with $\sigma^2 = 0$ having no effect.

Since the pixel-wise distributions are independent of each other, they can easily be combined into one multivariate Gaussian distribution for the whole image. This is called the *likelihood* for observation $\mathbf{y}$, given the image $\mathbf{x}$:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \prod_i \mathcal{N}(y_i; x_i, \sigma^2) \\ &= \mathcal{N}(\mathbf{y}; \mathbf{x}, \sigma^2 \mathbf{I}) \end{aligned} \tag{2.5}$$

In line with this assumption, the training data used to learn the model parameters later on will consist of natural images corrupted with artificial Gaussian noise of a certain *noise level* σ^2.

2.2.3 Blurred Images

If the perturbations applied to an image are not independent for each pixel, but can instead be described at each position as a combination of the content of the surrounding image region, the result is called a *blurred* image. Image blur usually occurs because some objects are out of focus, or because parts of the scene were moving relative to the imaging system while the image was taken. Both are obviously very common in photographs taken by hand.

Blur can also be caused by phenomena that refract and scatter the incoming light on its way, as is often the case in astronomical images. Currents of hot air and transparent objects, like windows, may have a similar effect in everyday scenes.

The discussion in this thesis will be limited to image blur that can be described by a single *blur kernel* $\mathbf{k}$ applied uniformly across the whole image, as is the case in the second and third image in Figure 2.3. A blur kernel in this sense is a linear filter with the additional properties that none of its elements can be negative, and all elements sum up to one. This is because it represents the effect of an underlying *point spread function*, which, simply speaking, models the distribution of light from a single point in the scene to multiple locations in the image. It is clear that no image location can receive a negative amount of light in this way, and that the total amount of light reaching the image remains the same.

Figure 2.3: Simulation of different types of blur applied to a natural image taken from the *Berkeley Segmentation Data Set* [1]. The original image *(left)* is subjected to motion blur with different kernels *(center left and center)*, to depth blur caused by camera focus *(center right)* and to radial blur caused by rotation around the camera axis *(right)*.

Because applying the blur kernel to an image is a *convolution* operation, its inverse, the corresponding *deblurring* operation, is also called *deconvolution* in this case. In this thesis it is additionally assumed that the blur kernel is known and does not have to be estimated from the observation, so the more specific term for the considered task would be *non-blind deconvolution*.

The assumption of spatially invariant blur across the whole image fits best for lateral movements of the camera during exposure, most commonly due to shaking. Although many blurry natural images also contain some rotational component [29], illustrated exaggeratedly in the fifth image in Figure 2.3, models based on the former assumption can still produce good results in practice.

In conjunction with some noise, which should not be neglected when looking at a blurred image, for each pixel y_i of the observed image $\mathbf{y}$ it is

$$y_i = \left(\sum_j K_{ij} x_j \right) + r \tag{2.6}$$

with a *blur matrix* $\mathbf{K}$ constructed from the given blur kernel $\mathbf{k}$ such that

$$\mathbf{K} \cdot \mathbf{x} = \mathbf{k} * \mathbf{x}$$

and pixel-independent Gaussian noise $r \sim \mathcal{N}(0, \sigma^2)$. It is easy to see that the simple Gaussian noise assumption from Equation (2.4) is obtained in the special case where $\mathbf{K} = \mathbf{I}$, that is, when there is no blur in effect. Note however that in general, pixels on the border of $\mathbf{y}$ will contain information from locations outside the observation "window". This means that the ground truth $\mathbf{x}_{\text{gt}}$ is actually larger than the observation $\mathbf{y}$, and the blur matrix $\mathbf{K}$ is not a square matrix. For the deblurring task, it means that the restored image also needs to be larger than the observation to be a full estimate of the ground truth.

When combining the pixel-wise terms from Equation (2.6) as it is done in the denoising case above, the result is the *likelihood* for observation $\mathbf{y}$, given an image $\mathbf{x}$,

in the form of a multivariate Gaussian distribution:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_i \mathcal{N}\left(y_i; \sum_j K_{ij} x_j, \sigma^2\right) = \mathcal{N}(\mathbf{y}; \mathbf{K}\mathbf{x}, \sigma^2 \mathbf{I}) \tag{2.7}$$

It should be mentioned that in the absence of noise and under certain boundary conditions, convolution with a blur kernel can directly be inverted and the original image fully recovered with the help of the *convolution theorem* [8]. Unfortunately, this solution produces very poor results even in the presence of weak noise, which can generally be assumed to affect all natural images. Different and more robust approaches must therefore to be employed to find a good estimate of the original image in real-world applications.

2.2.4 Finding the Restored Image

The noise and blur models introduced in the previous sections serve to describe the relation between a corrupted observation $\mathbf{y}$ and the underlying ground truth $\mathbf{x}$. But in order to find a good estimate $\hat{\mathbf{x}}$ for the ground truth when only the observation is known, which is the fundamental task that needs to be solved in image restoration, knowledge of $p(\mathbf{y} \mid \mathbf{x})$ is not sufficient.

When the effects of blur and noise are applied, many different images might lead to the same observation, and inverting the process without additional information is an ill-posed problem. In the denoising case, for example, $p(\mathbf{y} \mid \mathbf{y}) > p(\mathbf{y} \mid \mathbf{x})$ for any $\mathbf{x} \neq \mathbf{y}$ under the Gaussian assumptions stated above. Without further knowledge about the nature of $\mathbf{x}$, the most likely candidate for the ground truth is in fact the observation itself.

For solving the inverse problem of restoration, the real interest lies in the *posterior* distribution $p(\mathbf{x} \mid \mathbf{y})$, which, following *Bayes' rule* [5, § 1.2.3], can be expressed as

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x}) \cdot p(\mathbf{x})}{p(\mathbf{y})} \propto p(\mathbf{y} \mid \mathbf{x}) \cdot p(\mathbf{x}), \tag{2.8}$$

where $p(\mathbf{y})$ is constant in this setting because the observation $\mathbf{y}$ is given as input.

This means that in addition to the *likelihood* $p(\mathbf{y} \mid \mathbf{x})$ of $\mathbf{y}$ being a corrupted version of $\mathbf{x}$, it is necessary to model the *prior distribution* $p(\mathbf{x})$, or *prior* for short, which can be interpreted as the probability of $\mathbf{x}$ being a plausible natural image regardless of any observation $\mathbf{y}$. In order to maximize Equation (2.8), the restored image $\hat{\mathbf{x}}$ therefore has to strike a balance between maximizing *prior* and *likelihood*. Resolving the ambiguity by constraining $\hat{\mathbf{x}}$ in such a way is called *regularization*, and it requires the application of prior knowledge about what should be considered plausible natural images. A common way to encode this knowledge is covered in the following section.

2.3 Graphical Models

Graphical models, especially the undirected variant first introduced as a model for problems in statistical mechanics, have been studied and applied as image priors in

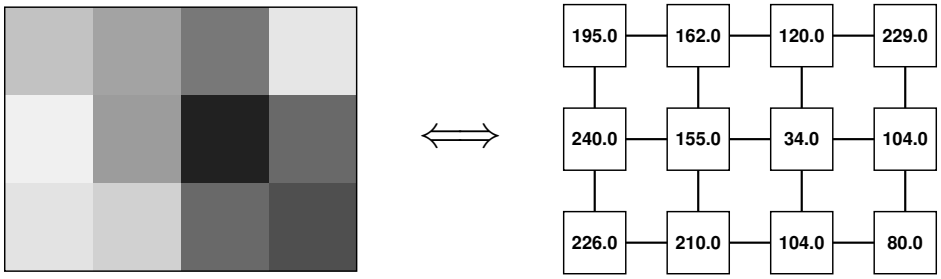


Figure 2.4: Example of the relation between the pixel intensities in an image and the configuration of the random variables situated on the nodes of a graphical model. For conventional 8-bit images, 0.0 corresponds to a black pixel and 255.0 to a white one.

computer vision for a long time [27, 21]. A graphical model describes the conditional interactions within a set of random variables that are arranged as nodes of a graph (V, E). In the image restoration context, each node in V represents one pixel x_i of an image $\mathbf{x}$, and the edges between them determine their conditional dependencies.

Each concrete image $\mathbf{x}$, by setting the random variables at each node to specific pixel values, describes a unique *configuration* of the model, as illustrated in Figure 2.4. The model on the other hand assigns a probability $p(\mathbf{x})$ to each configuration according to its definition and depending on a set of parameters.

2.3.1 Markov Random Fields

One family of undirected graphical models that has been particularly popular for computer vision tasks are *Markov random fields* [5, § 8.3]. They owe their name to the so-called *Markov property* that must be satisfied by all random variables, that is, all nodes in the model. This property is a *conditional independence* property and can be formulated as follows:

For any two subsets of nodes $A, B \subset V$ and a separating subset of nodes $C \subset V$ such that every path connecting a node from A to a node from B must pass through a node from C, it holds that

$$p(A \mid C) \cdot p(B \mid C) = p(A, B \mid C)\,,$$

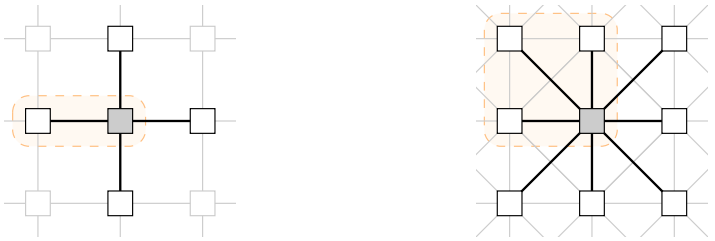
that is, A and B are conditionally independent given C. This is sometimes written in the short form $A \perp\!\!\!\perp B \mid C$ [5].

Perhaps a better way to think of it is in terms of the *Markov blanket* of a node x_i, which in Markov random fields is defined as the set of nodes directly connected to x_i via an edge in the graph:

$$\mathcal{M}(x_i) = \{x_j \in \mathbf{x} \mid (x_i, x_j) \in E\} \tag{2.9}$$

These nodes are also called the *neighbours* of x_i. The Markov property from above holds if x_i is conditionally independent of all other nodes in the graph given its Markov blanket, that is,

$$p(x_i \mid \{\mathbf{x}\} \setminus \{x_i\}) = p(x_i \mid \mathcal{M}(x_i))\,. \tag{2.10}$$



(a) Each node is connected to its 4 nearest neighbours, yielding maximal cliques of size 2.

(b) Each node is connected to its 8 nearest neighbours, yielding maximal cliques of size 2×2.

Figure 2.5: Difference between a *pairwise* Markov random field **(a)** and a *higher order* model **(b)**. One maximal clique is highlighted in each. Connecting each node to its 24 nearest neighbours would yield a higher order model with maximal cliques of size 3×3, and so forth.

As might be familiar from graph theory, a *clique* is any subset of nodes of a graph that is fully connected, or simply put, where there is an edge connecting each pair of nodes in the set. A *maximal clique* is one that is not contained in a larger clique. The vector of all pixels belonging to a clique c in the graph representation of $\mathbf{x}$ may be denoted by $\mathbf{x}_{(c)} \in \mathbb{R}^{|c|}$, and the set of all maximal cliques in the graph (V, E) is called $\mathcal{C}$.

According to the *Hammersley-Clifford theorem* [23], the joint probability of all variables in a Markov random field can be expressed as the product of *potential functions* φ_c over the cliques $c \in \mathcal{C}$,

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \varphi_c(\mathbf{x}_{(c)}) \,, \tag{2.11}$$

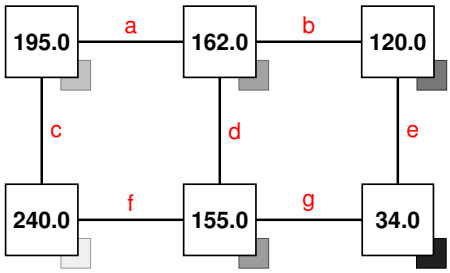
with the normalization constant

$$Z = \int \prod_{c \in \mathcal{C}} \varphi_c(\mathbf{x}_{(c)}) \, d\mathbf{x} \,, \tag{2.12}$$

which is called the *partition function*, ensuring that $p(\mathbf{x})$ is a valid probability distribution. This constant Z is defined as an integral over the range of all possible images $\mathbf{x}$, and as such cannot be evaluated in any reasonable amount of time even for discrete-valued images. Since the images handled here are continuous-valued, the partition function is generally *intractable*. There are different ways to deal with this problem, one of which will be presented later on in this chapter.

The most common, and most basic, form of a Markov random field only connects pixels which are directly horizontally or vertically adjacent, as illustrated in Figure 2.5a. This means that maximal cliques consist of only two nodes, leading to so-called *pairwise* potentials. As a result, the random field can only model the statistics of pairs of neighbouring pixels, which are assumed to be the same across all parts of the image, allowing the use of the same potential function φ for each clique.

It is also generally assumed that adjacent pixels are more likely to have similar than contrasting values, and φ should reflect that by awarding a better "score" to cliques



$$
\begin{aligned}
a &= \exp\left(-0.5 \cdot (195.0 - 162.0)^2\right) \approx 3.36 \cdot 10^{-237} \\
b &= \exp\left(-0.5 \cdot (162.0 - 120.0)^2\right) \approx 8.96 \cdot 10^{-384} \\
c &= \exp\left(-0.5 \cdot (195.0 - 240.0)^2\right) \approx 1.89 \cdot 10^{-440} \\
d &= \exp\left(-0.5 \cdot (162.0 - 155.0)^2\right) \approx 2.29 \cdot 10^{-11} \\
e &= \exp\left(-0.5 \cdot (120.0 - 34.0)^2\right) \approx 9.53 \cdot 10^{-1607} \\
f &= \exp\left(-0.5 \cdot (240.0 - 155.0)^2\right) \approx 1.29 \cdot 10^{-1569} \\
g &= \exp\left(-0.5 \cdot (155.0 - 34.0)^2\right) \approx 5.59 \cdot 10^{-3180}
\end{aligned}
$$

Figure 2.6: Example showing all clique potentials defined by the *pairwise Markov random field* model for a 2×3 image. The resulting numbers are very small and can not be properly represented by conventional floating point data types. Taking the product of all clique potentials in the image, as defined in Equation (2.11), exacerbates the problem, yielding essentially zero even for this minimal example.

with smaller pixel difference. This leads to the common form of the potential function as an exponential term

$$\varphi(\mathbf{x}_{(c)}) = \exp\big(-\rho(\mathbf{f}^\top \mathbf{x}_{(c)})\big)\,, \tag{2.13}$$

where $\mathbf{f}^\top = [-1, 1]$ is the *derivative filter* and ρ an even *penalty function*, for example the quadratic function $\rho(u) = \frac{1}{2}u^2$, penalizing inputs of large absolute value.

Together with Equation (2.11), the pairwise Markov random field thus defines a probability distribution

$$
\begin{aligned}
p(\mathbf{x}) &= \frac{1}{Z} \prod_{c \in \mathcal{C}} \exp\big(-\rho(\mathbf{f}^\top \mathbf{x}_{(c)})\big) \\
&= \frac{1}{Z} \prod_{c \in \mathcal{C}} \exp\left(-\frac{1}{2}\Big([-1, 1] \cdot \mathbf{x}_{(c)}\Big)^2\right)
\end{aligned} \tag{2.14}
$$

with the normalization constant becoming

$$Z = \int \prod_{c \in \mathcal{C}} \exp\left(-\frac{1}{2}\Big([-1, 1] \cdot \mathbf{x}_{(c)}\Big)^2\right) \, d\mathbf{x}\,.$$

For an image to have a high probability under such a random field model, no single clique may have a very low score, as each potential $\varphi(\mathbf{x}_{(c)})$ has a multiplicative influence on the result of Equation (2.11). This way, the local assumptions modelled by the potential function are enforced globally across the whole image.

Figure 2.6 shows an example of the clique potentials arising from this model for a small image. It also illustrates a central computational problem with the exponential function used in Equation (2.13), which is that it produces extremely small numbers when applied without any modification. A common way to address this problem is presented later on in this chapter.

Overall, the basic Markov random field architecture described so far proved to be a reliable and useful image model for many applications, but its capacity to reflect richer image statistics is obviously limited by its restriction to pairwise neighbourhood relations. To overcome this restriction, *higher order* Markov random fields

have been proposed which introduce additional edges in the graph, increasing the size of the Markov blankets and the maximal cliques. An illustration is given in Figure 2.5b. These additional connections allow relationships between multiple variables to be modelled by the potential functions, greatly increasing the expressiveness of the random field model.

2.3.2 Fields of Experts

A higher order random field with particular success in image restoration was introduced by Roth and Black [34, 35] and is called the *Field of Experts* model. It differs from the model defined in Equation (2.14) in two important aspects:

Firstly, since the cliques are no longer just pairs of nodes, but usually square image patches of size 3×3 up to 7×7, the simple derivative filter $\mathbf{f}$ has to be replaced by a linear filter of appropriate size. While $\mathbf{f}$ will still be written as a vector, it should be understood as the vector representation of a two-dimensional filter matching the dimensions of the maximal cliques. Furthermore, filters are constrained to have mean zero, like the derivative filter. This makes the model align well with the statistics of natural images [25].

Secondly, since many filters exist which could reasonably take the place of the derivative filter used before, and they all capture different properties of the image, a set of N filters $\mathbf{f}_i$ is used instead of just one. In order to model their respective responses, an equal number of different penalty functions ρ_i is needed. Following the original *Field of Experts* paper, the number N is usually set to the number of nodes in a maximal clique less one.

These pairs of a filter $\mathbf{f}_i$ and the corresponding penalty function ρ_i are then called *experts*, and the joint probability of the resulting model is defined as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \prod_{i=1}^{N} \exp\bigl(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\bigr) \tag{2.15}$$

with the normalization constant

$$Z = \int \prod_{c \in \mathcal{C}} \prod_{i=1}^{N} \exp\bigl(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\bigr)\, d\mathbf{x} \tag{2.16}$$

making it a proper distribution. Note that Z remains intractable.

It is easy to see that the pairwise model from Equation (2.14) is kept as a special case when $N = 1$, $\mathbf{f} = [-1, 1]^\top$, and nodes are only connected to their direct horizontal and vertical neighbours.

It is also clear that in contrast to the pairwise Markov random field, where the single filter and penalty function could be defined explicitly, higher order models have many more parameters that need to be adjusted. How this can be done with the help of machine learning and training data will be a central theme throughout Chapter 3.

Figure 2.7: A natural image from the *Berkeley Segmentation Data Set* [1] *(left)*, and the same image modified so as to achieve a higher probability under a Gaussian random field model *(right)*. Object contours and textured regions are smoothed very strongly to conform with the quadratic penalty function, losing some of the most important features of the image.

2.3.3 Robust Potentials

In the previous sections, the penalty functions ρ were introduced under no further assumptions than being even and returning larger values for arguments with larger absolute value. The only example mentioned so far was the quadratic penalty function $\rho(u) = \frac{1}{2}u^2$, which is a popular choice in many cases as it makes the potential functions $\varphi_i(\mathbf{x}_{(c)}) = \exp\left(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\right)$ take on a Gaussian form, which in turn leads to $p(\mathbf{x})$ resembling a multivariate Gaussian distribution

$$\begin{aligned} p(\mathbf{x}) &\propto \prod_{c \in \mathcal{C}} \prod_{i=1}^{N} \exp\left(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\right) \\ &\propto \prod_{c \in \mathcal{C}} \prod_{i=1}^{N} \mathcal{N}\left(\mathbf{f}_i^\top \mathbf{x}_{(c)}; 0, 1\right) \\ &\propto \mathcal{N}(\mathbf{x}; \mathbf{0}, \boldsymbol{\Sigma}_\mathbf{x}) \end{aligned} \tag{2.17}$$

where the sparse *precision matrix* $\boldsymbol{\Omega}_\mathbf{x} = \boldsymbol{\Sigma}_\mathbf{x}^{-1}$ can be easily computed [36].

As Section 2.3.4 will demonstrate, this is of great advantage during inference, but unfortunately, quadratic penalty functions also come with a great disadvantage.

While they enforce smoothness by assigning a very low probability to strong intensity jumps between neighbouring pixels, jumps like these regularly occur in almost all natural images, for example at object contours and in strongly textured regions. Penalizing them too heavily suppresses these image features in any output regularized with the image model in favour of smooth gradients as can be seen in Figure 2.7, which is obviously not the desired effect.

The filter responses of a set of zero-mean filters $\mathbf{f}_i$, as used in the *Field of Experts* model, follow a distribution that is very similar to that of pairwise pixel differences. One such distribution over a set of 68 natural images is shown in Figure 2.8, along with the potentials defined by four common penalty functions. The quadratic, or Gaussian potential can be seen to drastically underestimate the prevalence of stronger

filter responses, as does the *Laplacian* potential based on the ℓ_1-norm to a smaller degree.

This observation gives rise to the application of so-called *robust penalty functions* [6], a number of which are illustrated in Figure 2.9. In the first group, Figure 2.9a, the non-robust quadratic function is plotted against the linear *Laplacian* loss function and the quasi-linear *Huber* function. While the latter two are widely used as robust functions, they still put a large tax on stronger filter responses.

The second group, shown in Figure 2.9b, includes the logarithm-based *Lorentzian* penalty function and the root-based *hyper-Laplacian* [28] penalty function. Both of these still assign a larger penalty to inputs of larger absolute value, but are much more forgiving, or *robust*, towards outliers and resemble the actual distribution of the features in question much better (*cf.* Figure 2.8). The *Lorentzian* function in particular is smooth and easily differentiable everywhere, which makes it attractive for gradient-based optimization methods.

The functions in the last group, shown in Figure 2.9c, are robust by virtue of having an upper bound. This guarantees that no input can receive an excessively large penalty. Beyond a certain threshold, all filter responses are in fact assigned an equal, or nearly equal probabilty by potentials based on these penalty functions. Unfortunately, the two piecewise-defined functions can lead to difficulties during gradient-based optimization, as outside the central region their derivatives are zero and thus of no help when searching for an optimum.

However, the models considered in this thesis do not use any of the proposed functions. Instead, they exploit practically arbitrary penalty functions which will be introduced in the next chapter. Because the parameters defining their exact shape are learned, they nevertheless end up being robust as they adapt to model the statistics of the set of training images. Not being tied to any particular family of penalty functions in this way is one of the strong points of the approaches presented here.

2.3.4 Maximum A-Posteriori Inference

Having specified models for the prior probability $p(\mathbf{x})$ and the likelihood $p(\mathbf{y} \mid \mathbf{x})$ in Equations (2.7) and (2.15) respectively, one can now start looking for the image $\hat{\mathbf{x}}$ that maximizes the *posterior probability* from Equation (2.8),

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{y}) = \arg\max_{\mathbf{x}} \big(p(\mathbf{y} \mid \mathbf{x}) \cdot p(\mathbf{x})\big) \,.$$

This $\hat{\mathbf{x}}$ is called the *maximum a-posteriori* estimate.

Under the assumption that the penalty functions are quadratic, and thus the random field is Gaussian as in Equation (2.17), the posterior in combination with the Gaussian likelihood from Equation (2.7) takes on the form

$$p(\mathbf{x} \mid \mathbf{y}) \propto \mathcal{N}(\mathbf{y}; \mathbf{K}\mathbf{x}, \sigma^2\mathbf{I}) \cdot \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{\Omega}_\mathbf{x}^{-1}) \,,$$

which can be transformed using the *information parameterization* into

$$p(\mathbf{x} \mid \mathbf{y}) \propto \mathcal{N}_I\left(\mathbf{x}; \frac{\mathbf{K}^\top\mathbf{y}}{\sigma^2}, \frac{\mathbf{K}^\top\mathbf{K}}{\sigma^2}\right) \cdot \mathcal{N}_I\left(\mathbf{x}; \mathbf{0}, \mathbf{\Omega}_\mathbf{x}\right) \,, \tag{2.18}$$

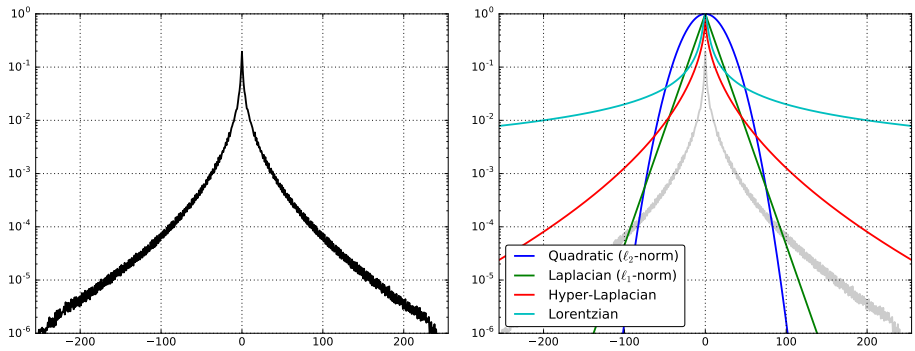
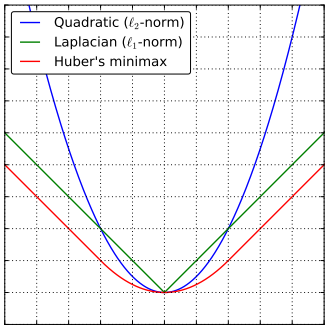
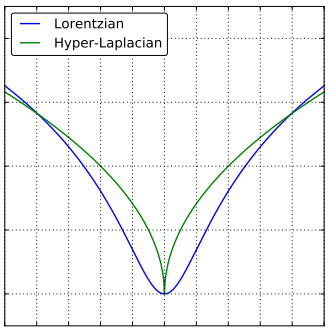


Figure 2.8: Distribution of filter responses for the DCT filter bank on a set of 68 natural images *(left)* compared to four common potential functions *(right)*. The robust *hyper-Laplacian* and *Lorentzian* potentials are clearly better suited for modelling a heavy-tailed distribution such as this.



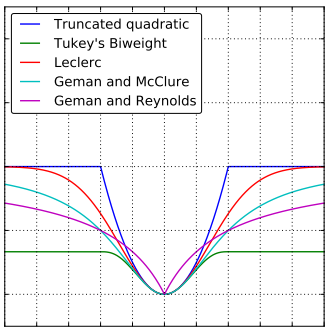
(a) Quadratic and linear penalty functions, in order of legend:

$$\rho(u) = \alpha \cdot u^2$$
$$\rho(u) = \alpha \cdot |u|$$
$$\rho(u) = \begin{cases} \frac{u^2}{2\epsilon} & \text{if } |u| < \epsilon \\ |u| - \frac{\epsilon}{2} & \text{otherwise} \end{cases}$$



(b) Logarithm and root based penalty functions, in order of legend:

$$\rho(u) = \alpha \cdot \ln\left(1 + (\beta u)^2\right)$$
$$\rho(u) = \alpha \cdot |u|^\gamma \text{ with } 0 < \gamma < 1$$



(c) Penalty functions with upper bound, in order of legend:

$$\rho(u) = \alpha \cdot \begin{cases} u^2 & \text{if } |u| < \epsilon \\ \epsilon^2 & \text{otherwise} \end{cases}$$
$$\rho(u) = \begin{cases} \frac{u^2}{\sigma^2} - \frac{u^4}{\sigma^4} + \frac{u^6}{3\sigma^6} & \text{if } |u| < \sigma \\ \frac{1}{3} & \text{otherwise} \end{cases}$$
$$\rho(u) = 1 - \exp\left(-\frac{u^2}{\sigma^2}\right)$$
$$\rho(u) = \frac{u^2}{1 + u^2}$$
$$\rho(u) = 1 - \frac{1}{1 + |u|}$$

Figure 2.9: Different families of penalty functions used to model image statistics

where both Gaussians describe distributions of $\mathbf{x}$.

According to [44], the product of two multivariate Gaussian distributions $\mathcal{N}_I(\mathbf{x}; \boldsymbol{\eta}_1, \boldsymbol{\Omega}_1)$ and $\mathcal{N}_I(\mathbf{x}; \boldsymbol{\eta}_2, \boldsymbol{\Omega}_2)$ in *information parameterization* is proportional to a third one $\mathcal{N}_I(\mathbf{x}; \boldsymbol{\eta}_1 + \boldsymbol{\eta}_2, \boldsymbol{\Omega}_1 + \boldsymbol{\Omega}_2)$. Applying this to Equation (2.18) gives

$$p(\mathbf{x} \mid \mathbf{y}) \propto \mathcal{N}_I\left(\mathbf{x}; \frac{\mathbf{K}^\top \mathbf{y}}{\sigma^2}, \frac{\mathbf{K}^\top \mathbf{K}}{\sigma^2} + \boldsymbol{\Omega}_\mathbf{x}\right), \tag{2.19}$$

which can be converted back to the *moment parameterization* as

$$p(\mathbf{x} \mid \mathbf{y}) \propto \mathcal{N}(\mathbf{x}; \boldsymbol{\Omega}_{\mathbf{x}|\mathbf{y}}^{-1} \boldsymbol{\eta}_{\mathbf{x}|\mathbf{y}}, \boldsymbol{\Omega}_{\mathbf{x}|\mathbf{y}}^{-1}) \tag{2.20}$$

with

$$\begin{aligned} \boldsymbol{\Omega}_{\mathbf{x}|\mathbf{y}}^{-1} &= \left(\frac{\mathbf{K}^\top \mathbf{K}}{\sigma^2} + \boldsymbol{\Omega}_x\right)^{-1}, \\ \boldsymbol{\eta}_{\mathbf{x}|\mathbf{y}} &= \frac{\mathbf{K}^\top \mathbf{y}}{\sigma^2}. \end{aligned}$$

Like for any other Gaussian, the maximizer $\hat{\mathbf{x}}$ of this distribution is its mean $\boldsymbol{\Omega}_{\mathbf{x}|\mathbf{y}}^{-1} \boldsymbol{\eta}_{\mathbf{x}|\mathbf{y}}$, which can be found directly by solving the linear equation system

$$\boldsymbol{\Omega}_{\mathbf{x}|\mathbf{y}} \cdot \hat{\mathbf{x}} = \boldsymbol{\eta}_{\mathbf{x}|\mathbf{y}}$$

for the vector $\hat{\mathbf{x}}$.

However, solving the analoguous inference problem exactly for a model with *robust* instead of quadratic penalty functions is not as straightforward. For this reason, a common strategy is to approximate the maximum a-posteriori solution with the help of iterative methods. But before looking for an estimate, it is useful to reformulate the problem in a slightly different manner.

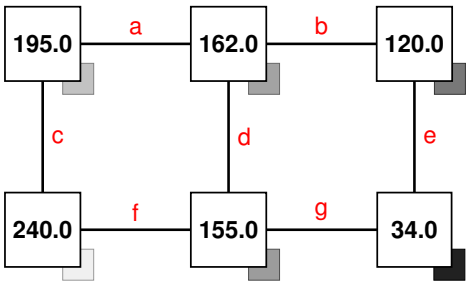
2.3.5 Energy Minimization

Because under a Gaussian noise assumption and with a *Field of Experts* prior, both the likelihood and the prior part in Equation (2.8) are exponential terms, the posterior probability is often rewritten as

$$\begin{aligned} p(\mathbf{x} \mid \mathbf{y}) &\propto p(\mathbf{y} \mid \mathbf{x}) \cdot p(\mathbf{x}) \\ &\propto \mathcal{N}(\mathbf{y}; \mathbf{K}\mathbf{x}, \sigma^2 \mathbf{I}) \cdot \frac{1}{Z} \prod_{c \in \mathcal{C}} \prod_{i=1}^{N} \exp\bigl(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\bigr) \\ &\propto \exp\bigl(-E(\mathbf{x} \mid \mathbf{y})\bigr) \end{aligned} \tag{2.21}$$

with an *energy term* of the form

$$E(\mathbf{x} \mid \mathbf{y}) \propto \frac{1}{2\sigma^2} \|\mathbf{K}\mathbf{x} - \mathbf{y}\|^2 + \sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}). \tag{2.22}$$



$$
\begin{aligned}
a &= 0.5 \cdot (195.0 - 162.0)^2 = 544.5 \\
b &= 0.5 \cdot (162.0 - 120.0)^2 = 882.0 \\
c &= 0.5 \cdot (195.0 - 240.0)^2 = 1012.5 \\
d &= 0.5 \cdot (162.0 - 155.0)^2 = 24.5 \\
e &= 0.5 \cdot (120.0 - 34.0)^2 = 3698.0 \\
f &= 0.5 \cdot (240.0 - 155.0)^2 = 3612.5 \\
g &= 0.5 \cdot (155.0 - 34.0)^2 = 7320.5
\end{aligned}
$$

Figure 2.10: Continuation of the example from Figure 2.6, this time showing the energy terms for all cliques of the *pairwise Markov random field* model for a 2×3 image. When the energy formulation is used, all resulting numbers can be represented by conventional data types. Taking the sum instead of the product over all cliques in an image further ensures that the numbers remain within a manageable range.

It is common to introduce an additional parameter λ, which can be called the *regularization weight*, determining the relative importance of likelihood and prior term. This parameter is multiplied to one of the terms, in this case the likelihood, and can be taken to subsume the constant factor $1/\sigma^2$ in order to simplify the equation:

$$
E(\mathbf{x} \mid \mathbf{y}) \propto \frac{\lambda}{2} \left\| \mathbf{K}\mathbf{x} - \mathbf{y} \right\|^2 + \sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}) \tag{2.23}
$$

This so-called *Gibbs energy* of the random field configuration specified by $\mathbf{x}$, which also originates from statistical physics, circumvents the problem of extremely small numbers which comes with exponential potential functions, as Figure 2.10 demonstrates. Nevertheless it retains its extremum in the same location, that is,

$$
\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{y}) = \arg\min_{\mathbf{x}} E(\mathbf{x} \mid \mathbf{y}) \,. \tag{2.24}
$$

Since the energy function of a non-Gaussian model is generally not convex, there is no closed-form solution to find its minimum, and iterative techniques need to be applied instead to find a good estimate. But because only the minimum is of interest here, and no other details of the underlying distributions are needed, constant multiplicative terms like the prefactor of the Gaussian distribution $p(\mathbf{y} \mid \mathbf{x})$ and the problematic normalization constant Z in $p(\mathbf{x})$ can safely be dropped in the energy formulation, making all further computations much easier.

Minimization of the energy function is commonly undertaken with iterative gradient-based methods, a popular one of which will be examined in Section 3.3.

3 Learned Inference

With the previous chapter having introduced a range of important general terminology and concepts, it is now time to have a look at the specific framework employed in this work to train an image restoration system, as well as the two approaches to be compared within that framework.

Recall from Section 2.1 that this thesis is following the *numerator layout* convention for all derivatives involving matrices and vectors.

3.1 Bi-level Optimization

Random field models have commonly been used in a *generative* setting, in which a prior like the *Field of Experts* considered here is tuned to model the statistics of natural images as well as possible, usually by maximizing the *likelihood* of a set of training images under the model. This results in a "general-purpose" prior that can be used in conjunction with many different applications. It also allows for drawing samples from the distribution of modelled images. A great disadvantage however is that the partition function Z from Equation (2.16) is usually intractable, which means that complicated and time-consuming approximation techniques are necessary for inference and training.

In [39], Samuel and Tappen proposed to instead train the model *discriminatively* for a specific application, such that the minimum energy solution for input images, obtained with some chosen inference method, is as close as possible to the desired ground truth. The use of ground truth data during training makes this a *supervised learning* strategy. There are two main advantages to the discriminative approach: by focusing on the energy term from Equation (2.23) instead of the full model as defined in Equation (2.15), the problematic *partition function* Z is eliminated from the procedure. Furthermore, optimizing the model to achieve good results when paired with a specific inference method has been shown to yield better performance than using a general-purpose model coupled with the same inference method [39, 2, 14].

Following this approach, the central problem of finding the optimal model parameters Θ^* can be formulated as a so-called *bi-level optimization* task

$$\Theta^* = \arg\min_{\Theta} \sum_{k=1}^{|\mathcal{S}|} \ell(\hat{\mathbf{x}}^k, \mathbf{x}_{\text{gt}}^k)\,, \tag{3.1}$$

where $\hat{\mathbf{x}}^k$ is the maximum a-posteriori solution

$$\hat{\mathbf{x}}^k = \arg\min_{\mathbf{x}} E(\mathbf{x} \mid \mathbf{y}^k; \Theta)\,. \tag{3.2}$$

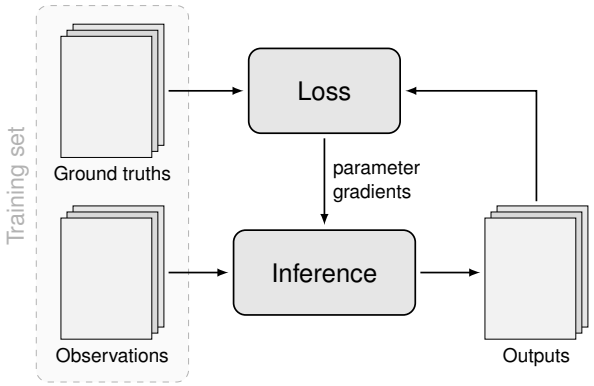


Figure 3.1: Illustration of the *bi-level optimization* framework. The model takes observations $\mathbf{y}^k$ from the training set $\mathcal{S}$ to produce outputs $\hat{\mathbf{x}}^k$, which are compared to the ground truths $\mathbf{x}^k$ by the loss function. The gradient of the loss function is then used to adjust the model parameters Θ in an iterative fashion and obtain better outputs.

$\mathcal{S}$ is a *training data set* consisting of pairs $(\mathbf{y}^k, \mathbf{x}^k_{\text{gt}})$ of observation and ground truth images, and $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})$ a *loss function* determining how well a model's prediction $\hat{\mathbf{x}}$ for an observation $\mathbf{y}$ resembles the corresponding ground truth $\mathbf{x}_{\text{gt}}$. An illustration of the resulting framework is provided in Figure 3.1.

Equation (3.1), minimizing the loss over the training set, is called the *upper level* of the bi-level framework and Equation (3.2), minimizing the energy term for a given observation, is called the *lower level*. Even though the lower level task is solved approximately via maximum a-posteriori estimation, which is likely to only arrive at a local optimum, the results reported in the above cited publications show that discriminative training still leads to very effective models.

In order to optimize the parameters in the upper level, Chen *et al.* [14] proposed applying the L-BFGS algorithm [31], which outperformed the original, less efficient optimization method used in [39]. L-BFGS is a sophisticated *quasi-Newton method* [9] which takes the first derivatives of the target function with respect to all parameters in order to construct an approximation of the inverse *Hessian matrix*, which in turn is used iteratively to find good search directions in the parameter space.

Implementations of L-BFGS are available in most scientific computing environments, so the inner workings of the method need not be discussed in detail. What need to be specified, however, are the first derivatives of the target function ℓ with respect to each element of Θ.

3.1.1 Loss Function

The loss function ℓ plays a critical role in this bi-level framework, acting as both an arbiter for the results of the lower level optimization and the target function to be minimized in the upper level.

One widely used measure of the difference between two vectors is their *Euclidean norm* or l^2*-norm*, which gives rise to the *mean squared error* (MSE) metric when

Figure 3.2: A natural image taken from the *Berkeley Segmentation Data Set* [1] once clean *(left)* and once with added Gaussian noise *(right, $\sigma^2 = 25.0$)*. The mean squared error between the two is 603.91, the peak signal-to-noise ratio 20.32.

applied to two $m \times n$ images $\hat{\mathbf{x}}$ and $\mathbf{x}_{\text{gt}}$:

$$\begin{aligned} \text{MSE}(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}}) &= \frac{1}{mn} \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|^2 \\ &= \frac{1}{mn} \sum_{i=1}^{mn} (\hat{x}_i - x_{\text{gt},i})^2 \,. \end{aligned} \tag{3.3}$$

Building on that, arguably the most popular difference measure in image restoration literature is the *peak signal-to-noise ratio* (PSNR) [40, 13]. It is defined as

$$\begin{aligned} \text{PSNR}(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}}) &= 10 \cdot \log_{10}\left(\frac{R^2}{\text{MSE}}\right) \\ &= 10 \cdot \log_{10}(R^2) - 10 \cdot \log_{10}(\text{MSE}) \\ &= 20 \cdot \log_{10}(R) - 10 \cdot \log_{10}(\text{MSE}) \,, \end{aligned} \tag{3.4}$$

where R is the *dynamic range* of the images, that is, the difference between the highest and lowest possible pixel value, which is usually 255.

With the MSE metric, lower values are obviously better when striving for similarity, as the minimal value of 0 marks identical images. For PSNR however, 0 is the worst case and the more similar the two images are, the higher their PSNR value goes, up to infinity for identical images. For this reason, PSNR is usually negated when used as a loss function to preserve the principle that the loss should be minimized:

$$\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}}) = -\text{PSNR}(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}}) \tag{3.5}$$

An example of two images and their resulting MSE and PSNR values can be seen in Figure 3.2.

Another interesting measure that has been proposed for image comparison is the *structural similarity index* (SSIM) [43]. While it is designed to better reflect human perception of structural differences in natural images, it is not as widely used as the PSNR, perhaps owing to its more complicated definition. It will not be considered it in this thesis.

As mentioned above, the optimization procedure needs the derivative of Equation (3.5) with respect to the model parameters Θ, which per chain rule is

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \Theta} = \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}} \cdot \frac{\partial \hat{\mathbf{x}}}{\partial \Theta}. \tag{3.6}$$

The inner derivative will be explored once the inference approaches used to obtain $\hat{\mathbf{x}}$ have been covered. For now, it will suffice to note that the outer derivative is

$$\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}} &= \frac{\partial}{\partial \hat{\mathbf{x}}} \left(-10 \cdot \log_{10} \left(\frac{R^2}{\text{MSE}} \right) \right) \\
&= \frac{\partial}{\partial \hat{\mathbf{x}}} \left(-\frac{20}{\log 10} \cdot \ln \left(\frac{R}{\sqrt{\text{MSE}}} \right) \right) \\
&= -\frac{20}{\ln 10} \cdot \frac{\partial}{\partial \hat{\mathbf{x}}} \ln \left(\frac{R \cdot \sqrt{mn}}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \right) \\
&= \frac{20}{\ln 10} \cdot \frac{\partial}{\partial \hat{\mathbf{x}}} \ln \left(\frac{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|}{R \cdot \sqrt{mn}} \right) \\
&= \frac{20}{\ln 10} \cdot \left(\frac{\partial}{\partial \hat{\mathbf{x}}} \ln \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\| - \frac{\partial}{\partial \hat{\mathbf{x}}} (R \cdot \sqrt{mn}) \right) \\
&= \frac{20}{\ln 10} \cdot \frac{1}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \cdot \frac{\partial}{\partial \hat{\mathbf{x}}} \|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\| \\
&= \frac{20}{\ln 10} \cdot \frac{1}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \cdot \frac{(\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}})^\top}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|} \\
&= \frac{20}{\ln 10} \cdot \frac{(\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}})^\top}{\|\hat{\mathbf{x}} - \mathbf{x}_{\text{gt}}\|^2}.
\end{aligned} \tag{3.7}$$

3.1.2 Training Data

For the training set $\mathcal{S}$, corresponding pairs $(\mathbf{y}^k, \mathbf{x}_{\text{gt}}^k)$ of corrupted observations and uncorrupted ground truths are needed. The standard way to obtain these is to take images $\mathbf{x}_{\text{gt}}^k$ from a data set which are assumed to be uncorrupted, and apply noise and blur according to the models from Equations (2.4) and (2.6) to get $\mathbf{y}^k$.

It is important to have a large enough set of training images, so that the learned model does not become biased towards features that were overrepresented in the training data by coincidence, or disregards features that were underrepresented. This phenomenon is called *overfitting* and gets more problematic with more expressive and flexible models, as these are especially prone to adapt to the peculiarities of their training data. To ensure that the learned model *generalizes* well to unseen data, the best solution is to have a training set that accurately represents the distribution of features in real world data. As a rule of thumb, having more non-redundant data generally helps in this regard, although Chen and Pock find in [13] that their denoising results do not significantly improve when going beyond 200 training image pairs.

Another matter is that of *quantization*. When noise and blur are applied exactly as stated above, they will naturally lead to real-valued pixel intensities in the corrupted image. The synthetically corrupted images thus differ from noisy and blurry images encountered in the real world, because the latter are usually constrained to

Figure 3.3: The 8 filters of the DCT filter base for filter size 3×3

integer values by the imaging process. This difference can be eliminated by applying *quantization* to the training images as well, that is, by rounding the pixel values in each image $\mathbf{y}^k$ to the nearest integer and truncating any values that fall outside the dynamic range $[0, 255]$:

$$\text{quant}(\mathbf{y}^k)_i = \min\left(255, \max\left(0, \left\lfloor y_i^k + \frac{1}{2} \right\rfloor\right)\right) \tag{3.8}$$

However, such a step is rarely applied to synthetically corrupted training and test data in existing literature, so to facilitate a direct comparison with other results it is better to use the real-valued images as they are.

3.1.3 Model Parameters

The parameters Θ have appeared in Equations (3.1) and (3.2), but it has not yet been addressed what they actually encompass and how they define their respective parts of the model. The following sections show how the linear filters $\mathbf{f}_i$ and the nonlinear functions ρ_i that make up the *Field of Experts* model can be specified in terms of weights vectors, and in the case of the functions, what their gradients look like.

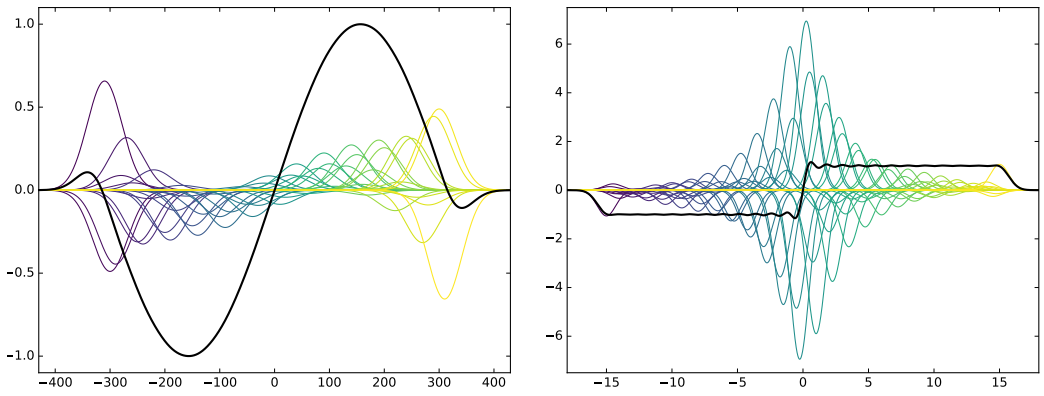
Parametrization of Linear Filters

It has been found in [25] that the sum of all elements of a meaningful image filter should be zero. Intuitively, this ensures that applying the filter to an image does not modify the mean intensity of the image, which is usually not a desired effect.

In order to still freely learn filter parameters, a practice applied in other works is to define a basis $\mathbf{B}$ of zero-sum filters that is then multiplied with a vector of weights $\tilde{\mathbf{f}}_i$ to obtain a new zero-sum filter $\mathbf{f}_i = \mathbf{B} \cdot \tilde{\mathbf{f}}_i$ [14, 40, 13]. The elements of these weights vectors $\tilde{\mathbf{f}}_i$ are the parameters that can then be set to arbitrary values without losing the zero-sum property.

For the basis $\mathbf{B}$, it seems best to choose the same modified 2D *DCT* filter basis used in the works cited above. To construct this basis for filters with k elements, the second to k^{th} columns of the identity matrix $\mathbf{I}^{k \times k}$ are taken as individual vectors and the inverse *discrete cosine transform* is applied to each of them. This yields $k - 1$ orthogonal, zero-sum basis vectors of length k, so each weights vector $\tilde{\mathbf{f}}_i$ must have $k - 1$ elements. The resulting filters for $k = 9$, which are used in models with 3×3 cliques, can be seen in Figure 3.3.

Since the filters $\mathbf{f}_i$ do not appear directly in the inference procedures described later on, the discussion of their gradients will be postponed until the concrete inference formulations have been stated.



(a) $f(x) = \sin(x/100)$ as the sum of 63 Gaussians on the interval $[-310, 310]$

(b) $f(x) = \text{signum}(x)$ as the sum of 121 Gaussians on the interval $[-15, 15]$

Figure 3.4: Radial basis function mixtures approximating two functions. The coloured curves are the Gaussian basis functions and the black curve is their sum. Although the signum function can not be approximated perfectly, the result is smooth and differentiable.

Parametrization of Nonlinear Functions

In order to learn practically arbitrary nonlinear functions, a representation that depends on a manageable number of parameters is needed, which should additionally be easily differentiable. One solution for this problem is given in the form of Gaussian *radial basis function mixtures* [33, 18, 24], in short RBF mixtures, which compose the function value as a sum of K Gaussian distributions,

$$\rho(x) = \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right), \tag{3.9}$$

with weights and means vectors $\boldsymbol{\omega}, \boldsymbol{\mu} \in \mathbb{R}^K$ and a shared precision parameter β governing the range of influence of each Gaussian in the mixture. An example of two functions represented by such radial basis function mixtures is given in Figure 3.4.

It is easy to see that the derivative of Equation (3.9) with respect to weight ω_i is

$$\frac{\partial \rho(x)}{\partial \omega_i} = \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right), \tag{3.10}$$

while the derivative with respect to the input x is

$$\begin{aligned}
\frac{\partial \rho(x)}{\partial x} &= \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right) \cdot \left(-\frac{1}{2} \cdot \beta \cdot 2 \cdot (x - \mu_i)\right) \\
&= -\beta \cdot \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right) \cdot (x - \mu_i) \\
&=: \rho'(x)\,.
\end{aligned} \tag{3.11}$$

Since these nonlinear functions will be applied separately to each element of some

vector in the final formulation of the inference procedures, it is useful to define

$$\rho(\mathbf{x}) \coloneqq \begin{bmatrix} \rho(x_1) \\ \vdots \\ \rho(x_M) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_1 - \mu_i\|^2\right) \\ \vdots \\ \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_M - \mu_i\|^2\right) \end{bmatrix} \tag{3.12}$$

and

$$\rho'(\mathbf{x}) \coloneqq \begin{bmatrix} \rho'(x_1) \\ \vdots \\ \rho'(x_M) \end{bmatrix} = \begin{bmatrix} -\beta \cdot \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_1 - \mu_i\|^2\right) \cdot (x_1 - \mu_i) \\ \vdots \\ -\beta \cdot \sum_{i=1}^{K} \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_M - \mu_i\|^2\right) \cdot (x_M - \mu_i) \end{bmatrix} \tag{3.13}$$

for vector-valued arguments $\mathbf{x} \in \mathbb{R}^M$.

Using Equation (3.10) and following the *numerator layout* convention as described in Section 2.1, for vector-valued arguments $\mathbf{x} \in \mathbb{R}^M$ it is

$$\begin{aligned} \frac{\partial \rho(\mathbf{x})}{\partial \boldsymbol{\omega}} &= \mathbf{G} \in \mathbb{R}^{M \times K} \\ \text{with} \quad G_{ij} &= \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_i - \mu_j\|^2\right) \end{aligned} \tag{3.14}$$

and similarly from Equation (3.11) it follows

$$\begin{aligned} \frac{\partial \rho(\mathbf{x})}{\partial \mathbf{x}} &= \mathbf{D} \in \mathbb{R}^{M \times M} \\ \text{with } D_{ij} &= \frac{\partial \rho(x_i)}{\partial x_j} \\ &= \begin{cases} -\beta \cdot \sum_{k=1}^{K} \omega_k \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x_i - \mu_k\|^2\right) \cdot (x_i - \mu_k), & \text{if } i = j \\ 0, & \text{otherwise,} \end{cases} \end{aligned}$$

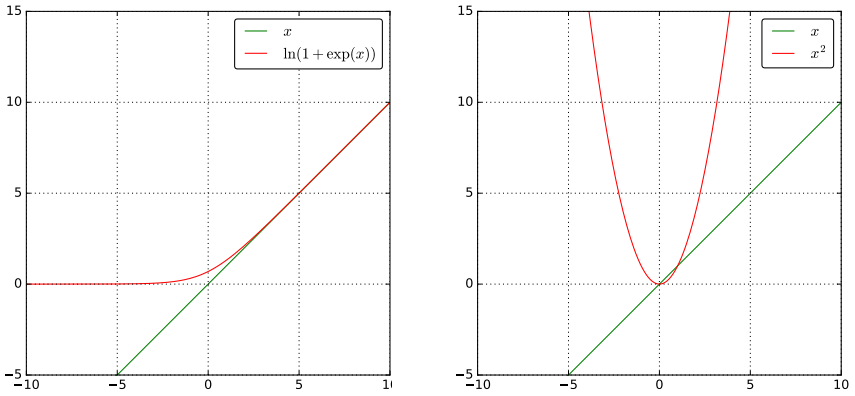
which together with the definition from Equation (3.13) yields

$$\frac{\partial \rho(\mathbf{x})}{\partial \mathbf{x}} = \operatorname{diag}\left\{\rho'(\mathbf{x})\right\}. \tag{3.15}$$

The result is a diagonal matrix because of the way ρ is applied independently to each element of $\mathbf{x}$, making the off-diagonal partial derivatives 0.

Constraining Functions to Positive Values

At a later point it will be necessary to approximate functions under the constraint that their values must not be negative. This can relatively easily be achieved with Gaussian *radial basis functions* by replacing the weights with an expression that is



(a) ln and exp **(b)** Naive squaring

Figure 3.5: Two ways of constraining the value of a parameter to be positive while staying differentiable. The one on the left is a much better approximation of the original parameter for positive values, but its derivative becomes practically zero for negative values, which can lead to problems during gradient-based optimization.

always positive, since the unweighted Gaussian components already adhere to the constraint.

One way this can be done is to substitute the weight ω_i in Equation (3.9) by $\ln\left(1+\exp\left(\omega_i\right)\right)$. This term is practically equal to ω_i for large values of ω_i, but converges to 0 as ω_i goes to $-\infty$ (see Figure 3.5a). The constrained *radial basis function* mixture then looks like

$$\rho(x) = \sum_{i=1}^{K} \ln\left(1+\exp\left(\omega_i\right)\right) \cdot \exp\left(-\frac{1}{2}\cdot\beta\cdot\|x-\mu_i\|^2\right), \tag{3.16}$$

while

$$\frac{\partial\rho(x)}{\partial\omega_i} = -\frac{\exp\left(\omega_i\right)}{1+\exp\left(\omega_i\right)} \cdot \exp\left(-\frac{1}{2}\cdot\beta\cdot\|x-\mu_i\|^2\right) \tag{3.17}$$

and

$$\frac{\partial\rho(x)}{\partial x} = -\beta\cdot\sum_{i=1}^{K} \ln\left(1+\exp\left(\omega_i\right)\right) \cdot \exp\left(-\frac{1}{2}\cdot\beta\cdot\|x-\mu_i\|^2\right)\cdot(x-\mu_i)\,. \tag{3.18}$$

Unfortunately, this approach turns out to have adverse effects on training. The factor $\exp\left(\omega_i\right)/\left(1+\exp\left(\omega_i\right)\right)$ in the weights gradient converges to 0 as the weight ω_i goes to $-\infty$, making the gradient disappear. As a result, once a weight has reached a low enough value, it is stuck there since movement along the gradient can only happen in such small steps as to be practically negligible.

A different option is to just substitute ω_i with ω_i^2, which is also guaranteed to be positive for $\omega_i \neq 0$ (see Figure 3.5b). The constrained function is then simply

$$\rho(x) = \sum_{i=1}^{K} \omega_i^2 \cdot \exp\left(-\frac{1}{2}\cdot\beta\cdot\|x-\mu_i\|^2\right) \tag{3.19}$$

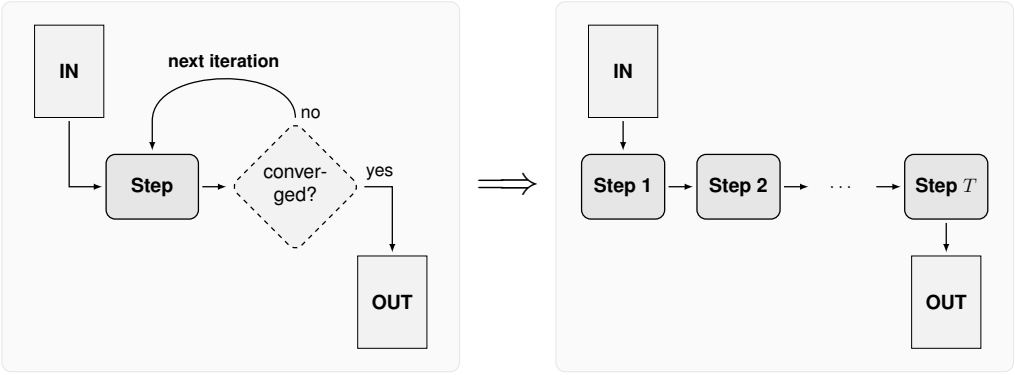


Figure 3.6: Iterative inference, repeated until convergence *(left)* and truncated inference, repeated for a fixed number of steps T *(right)*. The process of converting the former to the latter is sometimes called *unrolling*, as the loop is broken up into a linear succession of steps.

with derivatives

$$\frac{\partial \rho(x)}{\partial \omega_i} = 2 \cdot \omega_i \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right) \tag{3.20}$$

and

$$\frac{\partial \rho(x)}{\partial x} = -\beta \cdot \sum_{i=1}^{K} \omega_i^2 \cdot \exp\left(-\frac{1}{2} \cdot \beta \cdot \|x - \mu_i\|^2\right) \cdot (x - \mu_i)\,. \tag{3.21}$$

It is apparent that the gradients again break down when ω_i is exactly 0. In practice however, all computations use floating point arithmetic and this case is almost sure not to happen by accident. The quadratic substitution therefore lead to better trainable positive-constrained functions in preliminary experiments, even though the former variant does look more robust on paper.

In both cases, the derivatives for vector-valued arguments can be constructed like in Equations (3.14) and (3.15) above, but using the modified scalar derivatives as building blocks.

3.2 Truncated Optimization

Although the bi-level framework introduced in Section 3.1 simplifies training by removing the intractable partition function, it still requires a solution of the maximum a-posteriori inference problem in the lower level. Unfortunately, the iterative inference methods that are generally suited for this task may take a great number of steps until they converge at an optimum, and it is difficult to say in advance how long this process will take. Because of this, there is also no clear closed-form solution for the derivative of the output $\hat{\mathbf{x}}$ with respect to the model parameters Θ. In addition, the reached optimum is still likely to be a local one dependent on initialization, so the solution can not be expected to be exact in the first place.

Following this train of thought, Barbu proposed a *truncated optimization* scheme for the lower level in [2], limiting the iterations of the inference procedure to a small

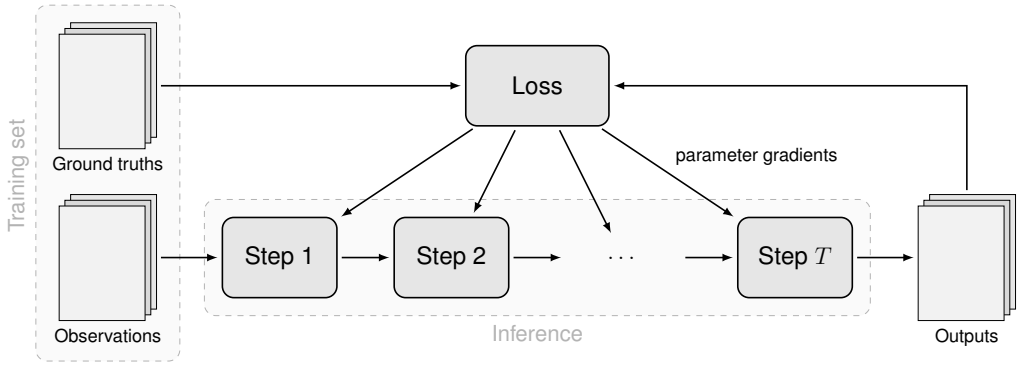


Figure 3.7: The framework from Figure 3.1 with truncated number of inference steps.

number of steps T by unrolling it as illustrated in Figure 3.6. This obviously results in a less accurate estimate of the energy minimum, but by training the whole model so as to minimize the total loss over a training set, this estimation error can be compensated by a "biased" model. While the resulting model performs on par with non-truncated architectures, its great advantage lies in the much faster computation time both during inference and learning.

The *bi-level optimization task* formulated in Equations (3.1) and (3.2) is modified for this truncated model and can be written as

$$\Theta^* = \arg\min_{\Theta} \sum_{k=1}^{|\mathcal{S}|} \ell(\hat{\mathbf{x}}^k, \mathbf{x}_{\mathrm{gt}}^k)\,, \tag{3.22}$$

with $\hat{\mathbf{x}}^k$ now being the result of the truncated maximum a-posteriori inference

$$\hat{\mathbf{x}}^k = \hat{\mathbf{x}}_T^k(\hat{\mathbf{x}}_{T-1}^k(\cdots \hat{\mathbf{x}}_1^k(\hat{\mathbf{x}}_0; \mathbf{y}^k, \Theta) \cdots ; \mathbf{y}^k, \Theta); \mathbf{y}^k, \Theta)\,. \tag{3.23}$$

A related concept is that of *unfolding* a *recurrent neural network* [7], sometimes also called *unrolling*, which goes back all the way to the inventors of the *backpropagation* algorithm [37]. In that case, some nodes in the network feed their output back as input for themselves or predecessors, creating a loop which bears some similarity to an iterative procedure. By unfolding the loop into a fixed number of conventional nodes, usual neural network algorithms can be applied in a setting they would otherwise not support.

An illustration of the *truncated bi-level optimization framework* is provided in Figure 3.7. Later works developed Barbu's approach from [2] further by allowing each of the limited number of inference steps to have a separate set of model parameters Θ_t [40, 13]. The truncated optimization then resembles a *cascade* or *pipeline* of specialized models, where each step refines the output of its predecessor. The speed of inference is not diminished by this modification, as the same amount of computation is needed to generate an output. Model capacity however is clearly increased with the gained flexibility, as is shown by the improved results in several image restoration applications.

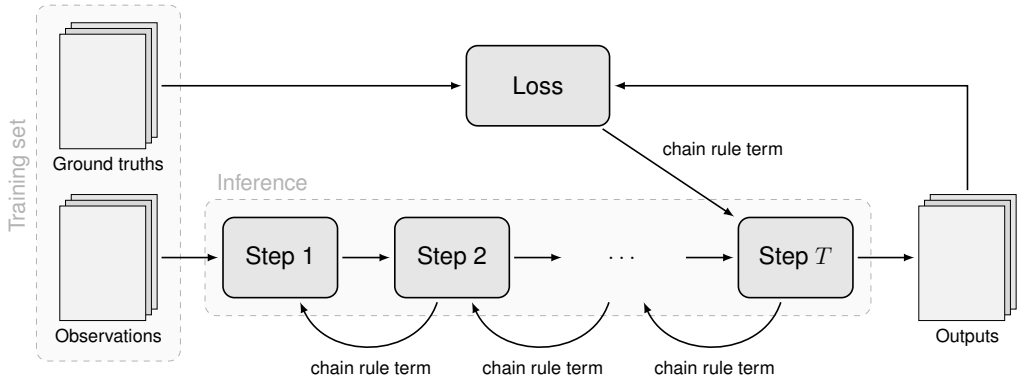


Figure 3.8: Application of the backpropagation algorithm to the truncated model from Figure 3.7. The *forward pass* begins at the first step and ends at the loss function, while the *backward pass* starts with the calculated loss and propagates its derivatives all the way back to the first step, making repeated use of the chain rule.

3.2.1 Computing Gradients with Backpropagation

A succession of T steps in this cascade model works such that the output of one step constitutes the input for the next one, until the final step produces the output of the overall model. Viewed the other way around, the output of each step depends only on that step's parameters and the output of the previous one, except for the first step which takes as input an initial guess $\hat{\mathbf{x}}_0$:

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}_T(\hat{\mathbf{x}}_{T-1}(\cdots \hat{\mathbf{x}}_1(\hat{\mathbf{x}}_0; \mathbf{y}, \Theta_1) \cdots ; \mathbf{y}, \Theta_{T-1}); \mathbf{y}, \Theta_T) \tag{3.24}$$

This chain can also be compared to a simple *feed-forward neural network* [5, § 5.1]. The standard technique to calculate parameter gradients for the different nodes in such a network is the *backpropagation* algorithm [37][5, § 5.3] mentioned above. It works by calculating the error of the output $\hat{\mathbf{x}}$ produced by a *forward pass* of the network, compared to the desired output $\mathbf{x}_{\text{gt}}$ via a loss function, and then performing a *backward pass* to compute the derivatives in each node using the chain rule. During this backward pass, the output error is propagated back through the network and parameters can be adjusted depending on how they contributed to this error. An important prerequisite for this is that the operations performed in each node must be differentiable.

Applied to the formulation in Equation (3.24), the derivative of the loss with respect to the parameters Θ_t of the t^{th} step is

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \Theta_t} = \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \Theta_t}\,, \tag{3.25}$$

where the *chain rule term*

$$\hat{\mathbf{c}}_t^\top := \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} = \hat{\mathbf{c}}_{t+1}^\top \cdot \frac{\partial \hat{\mathbf{x}}_{t+1}}{\partial \hat{\mathbf{x}}_t} = \hat{\mathbf{c}}_{t+2}^\top \cdot \frac{\partial \hat{\mathbf{x}}_{t+2}}{\partial \hat{\mathbf{x}}_{t+1}} \cdot \frac{\partial \hat{\mathbf{x}}_{t+1}}{\partial \hat{\mathbf{x}}_t} = \dots \tag{3.26}$$

can be computed iteratively by starting with the last step, which gets its chain rule term

$$\hat{\mathbf{c}}_T^\top = \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}}$$

directly from the derivative of the loss function in Equation (3.6).

Since the update functions $\hat{\mathbf{x}}_t(\hat{\mathbf{x}}_{t-1}; \mathbf{y}, \Theta_t)$ of the iterative inference approaches considered in this thesis are all differentiable with respect to $\hat{\mathbf{x}}_{t-1}$, this yields an efficient way to obtain the parameter gradients for all steps in one go. As a result, it is possible to perform *end-to-end training*, which means that all parts of the system from input to output can be trained simultaneously.

The interactions between the steps of a truncated inference model when applying this backpropagation scheme are illustrated in Figure 3.8.

3.2.2 Greedy and Joint Training

As seen above, the backpropagation algorithm can produce derivatives of the loss function with respect to each parameter in the whole model. These can then be concatenated into one large gradient vector $\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})/\partial \Theta$, averaged over all image pairs in the training set, and passed to the L-BFGS solver to update the parameters all at once, for as many iterations as required or feasible. Training all parameters at the same time like this is called *joint training*.

The obvious advantage is that parameters in earlier steps can be adjusted so as to produce the most useful output for the following steps. On the other hand, the number of derivatives that have to be computed for each iteration of the L-BFGS solver can become very large, which may lead to long computation times.

An alternative way to train the model is to start with only one step and optimize its parameters on their own. Once they have converged, a second step is added and trained on its own, keeping the parameters of the first step fixed. This can be continued until the model has the desired number of steps, but only the parameters of the last step in the chain are optimized at any time. This approach is called *greedy training*, since the output of each step is maximized without considering its interaction with later steps.

Training a model in this way has the advantage of breaking the optimization down into more manageable chunks. It also means that the model output when stopping after any individual step will be optimized in this greedy manner, so the trained model automatically contains trained versions of each "shorter" model. With joint training, this is not the case, as intermediate steps are optimized to support their successors and not to produce a good output. Lastly, the model obtained with greedy training can of course be used as initialization for further joint training, which can be seen as fine-tuning the interactions of the steps.

Both Schmidt and Roth [40] and Chen and Pock [13] find in their respective experiments that joint training without a greedy initialization produces measurably worse results than greedy training alone, whereas tuning a greedily trained model by additional joint training only marginally improves results. For this reason, the greedy approach is preferred during the experiments conducted for this thesis. The derivations in the following sections nonetheless cover both approaches. The main difference in this regard is the need for the input gradients $\partial \hat{\mathbf{x}}_t / \partial \hat{\mathbf{x}}_{t-1}$ when training jointly.

3.3 Inference via Gradient Descent

After introducing all of these definitions and concepts to properly set the context, everything is in place to have an in-depth look at the two concrete inference approaches mentioned in the first chapter, the comparison of which is at the heart of this work.

First to be examined is the gradient descent approach for minimizing the energy function defined in Section 2.3.5. Its suitability for the truncated optimization framework has been shown repeatedly [2, 13], but no results have yet been published for the deblurring problem.

Although the derivations for this approach can already be found in a similar form in [13] and partly in [40], they are reproduced here, with adapted notation, as they play a very crucial role in solving the optimization task.

3.3.1 Motivation

As could be seen in Section 2.3.5, a maximum a-posteriori estimate of the restored image $\hat{\mathbf{x}}$ can be found by minimizing the energy term from Equation (2.23). Because there is no closed-form solution for this minimum, the most direct strategy is searching for it with an iterative gradient-based procedure.

Perhaps the most obvious such procedure, given a previous estimate $\hat{\mathbf{x}}_{t-1}$, is to move through the search space by taking steps in the direction of steepest descent, indicated by the gradient of the target function E at the current position. This results in an iteration step

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} - \alpha_t \cdot \left(\frac{\partial E(\hat{\mathbf{x}}_{t-1} \mid \mathbf{y})}{\partial \hat{\mathbf{x}}_{t-1}} \right)^\top \tag{3.27}$$

with α_t determining the size of the step to be taken.

This approach is called *gradient descent* or *steepest descent*. In its elegant simplicity, it is one of the classical methods of optimization, going all the way back to the work of the French mathematician Cauchy [10]. All it requires in order to be applied is the gradient of the energy function.

3.3.2 Energy Gradient

Differentiating the energy term with respect to the previous estimate $\hat{\mathbf{x}}_{t-1}$, keeping in mind that each iteration t gets its own set of parameters, yields

$$\begin{aligned}
\frac{\partial E(\hat{\mathbf{x}}_{t-1} \mid \mathbf{y})}{\partial \hat{\mathbf{x}}_{t-1}} &= \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \left(\frac{\lambda_t}{2} \left\| \mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y} \right\|^2 + \sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \rho_{ti}(\mathbf{f}_{ti}^\top \hat{\mathbf{x}}_{(c)t-1}) \right) \\
&= \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \left(\sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \rho_{ti}(\mathbf{f}_{ti}^\top \hat{\mathbf{x}}_{(c)t-1}) \right) + \lambda_t \cdot \left(\mathbf{K}^\top (\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right)^\top \\
&= \sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \rho'_{ti}(\mathbf{f}_{tic}^\top \hat{\mathbf{x}}_{t-1}) \cdot \mathbf{f}_{tic}^\top + \lambda_t \cdot \left(\mathbf{K}^\top (\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right)^\top
\end{aligned}$$

with

$$\mathbf{f}_{tic}^{\top}\hat{\mathbf{x}}_{t-1} = \mathbf{f}_{ti}^{\top}\mathbf{M}_c\hat{\mathbf{x}}_{t-1} = \mathbf{f}_{ti}^{\top}\hat{\mathbf{x}}_{(c)t-1} \tag{3.28}$$

via an easy to construct masking matrix $\mathbf{M}_c$ that reduces the vector representation $\hat{\mathbf{x}}_{t-1}$ of an image to the clique vector $\hat{\mathbf{x}}_{(c)t-1}$ of clique c. The clique-specific filter $\mathbf{f}_{tic}^{\top}$ can therefore be constructed as $\mathbf{f}_{tic}^{\top} = \mathbf{f}_{ti}^{\top} \cdot \mathbf{M}_c$.

Since from here on out, only the derivatives ρ'_{ti} of the penalty functions will be used and the original functions ρ_{ti} will not appear again, the parameterized functions ρ_{ti} can just be set to model the derivatives directly, yielding the modified energy derivative

$$\frac{\partial E(\hat{\mathbf{x}}_{t-1} \mid \mathbf{y})}{\partial \hat{\mathbf{x}}_{t-1}} = \sum_{c\in\mathcal{C}}\sum_{i=1}^{N} \rho_{ti}(\mathbf{f}_{tic}^{\top}\hat{\mathbf{x}}_{t-1}) \cdot \mathbf{f}_{tic}^{\top} + \lambda_t \cdot \left(\mathbf{K}^{\top}(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y})\right)^{\top}. \tag{3.29}$$

To eliminate the summation over all cliques from the equation, a common trick [14, 40] is to introduce a *filter matrix* $\mathbf{F}_{ti}$ condensing all clique-specific filters $\mathbf{f}_{tic}^{\top}$ as

$$\mathbf{F}_{ti} = \begin{bmatrix} \mathbf{f}_{tic_1}^{\top} \\ \vdots \\ \mathbf{f}_{tic_{\text{end}}}^{\top} \end{bmatrix}, \tag{3.30}$$

which helps to reduce the double sum as follows

$$\begin{aligned} \sum_{c\in\mathcal{C}}\sum_{i=1}^{N} \rho_{ti}(\mathbf{f}_{tic}^{\top}\hat{\mathbf{x}}_{t-1}) \cdot \mathbf{f}_{tic}^{\top} &= \sum_{i=1}^{N}\sum_{c\in\mathcal{C}} \rho_{ti}(\mathbf{f}_{tic}^{\top}\hat{\mathbf{x}}_{t-1}) \cdot \mathbf{f}_{tic}^{\top} \\ &= \sum_{i=1}^{N} \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^{\top} \cdot \mathbf{F}_{ti} \\ &= \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^{\top} \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\right)^{\top}, \end{aligned} \tag{3.31}$$

recalling from Section 3.1.3 that ρ_{ti} is applied element-wise to vector-valued arguments by definition.

3.3.3 Update Formula

Putting all of this together and plugging it into Equation (3.27) leads to an update formula for one iteration of gradient descent of the form

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} - \alpha_t \cdot \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^{\top} \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) + \lambda_t \cdot \mathbf{K}^{\top}(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y})\right).$$

For the initial estimate $\hat{\mathbf{x}}_0$, one can simply use the observation $\mathbf{y}$ as the best available guess. The regularization weight λ_t that has been introduced from Equation (2.23) is handled as an additional model parameter and will be learned alongside the filters and functions.

Since every additive term inside the energy gradient is either multiplied by λ_t or by one of the ρ_{ti}, which are freely scalable and will be learned from training data, the step size α_t is in fact redundant and can simply be dropped.

Furthermore, it would not make sense to have the regularization weight be negative or zero. To ensure that $\lambda_t > 0$ at all times, it will be constrained as λ_t^+ in the same way that was considered to constrain the function weights to be positive in Section 3.1.3, that is,

$$\lambda_t^+ = \ln\left(1 + \exp \lambda_t\right). \tag{3.32}$$

For an illustration, see Figure 3.5a.

With these changes, the update formula for the t^{th} gradient descent step is obtained as

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) + \lambda_t^+ \cdot \mathbf{K}^\top(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right). \tag{3.33}$$

3.3.4 Derivative of the Loss Function

Recall from Section 3.2 that back-propagation is utilized to learn the parameters of each step according to an overall loss function ℓ (*cf.* Equation (3.5)), so it is necessary to compute the derivative of each parameter with respect to ℓ. As per chain rule, the general derivative of ℓ with relation to any variable v_t occurring in $\hat{\mathbf{x}}_t$ from Equation (3.33) is

$$\begin{aligned} \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial v_t} &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial v_t}, \\ \text{where} \quad \hat{\mathbf{c}}_t^\top &= \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \end{aligned} \tag{3.34}$$

is the outer derivative or *chain rule term*, passed back by the next step in the model or, in case of the last step, directly from the loss function as derived in Equation (3.7).

3.3.5 Input Gradient

Since the whole backpropagation approach for joint training hinges on using the chain rule, the step update formula from Equation (3.33) must be differentiated not only with respect to the parameters that are to be learned, but also with relation to the previous step's output $\hat{\mathbf{x}}_{t-1}$:

$$\begin{aligned} \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_{t-1}} &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \hat{\mathbf{x}}_{t-1}} \\ &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \cdot \left(\hat{\mathbf{x}}_{t-1} - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) + \lambda_t^+ \cdot \mathbf{K}^\top(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right) \right) \\ &= \hat{\mathbf{c}}_t^\top \cdot \left(\mathbf{I} - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \hat{\mathbf{x}}_{t-1}} + \lambda_t^+ \cdot \mathbf{K}^\top\mathbf{K} \right) \right) \end{aligned}$$

$$
\begin{aligned}
&= \hat{\mathbf{c}}_t^\top \cdot \left(\mathbf{I} - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\left\{ \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \right\} \cdot \mathbf{F}_{ti} + \lambda_t^+ \cdot \mathbf{K}^\top \mathbf{K} \right) \right) \\
&= \hat{\mathbf{c}}_t^\top - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\left\{ \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \right\} \cdot \mathbf{F}_{ti}\hat{\mathbf{c}}_t + \lambda_t^+ \cdot \mathbf{K}^\top \mathbf{K}\hat{\mathbf{c}}_t \right)^\top
\end{aligned} \tag{3.35}
$$

This derivative is then passed back to the previous step to serve as $\hat{\mathbf{c}}_{t-1}^\top$ as described in Section 3.2.1.

3.3.6 Parameter Gradients

The following passages show how the loss function $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})$ can be differentiated with respect to the model parameters λ_t, $\boldsymbol{\omega}_{ti}$ and $\tilde{\mathbf{f}}_{ti}$ of the t^{th} step.

Regularization Weight

The derivative of the loss with respect to the regularization weight λ_t is then given as

$$
\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \lambda_t} &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \lambda_t} \\
&= \hat{\mathbf{c}}_t^\top \cdot \left(\frac{\partial \lambda_t^+}{\partial \lambda_t} \cdot \mathbf{K}^\top (\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right) \\
&= \hat{\mathbf{c}}_t^\top \cdot \left(-\frac{\exp \lambda_t}{1 + \exp \lambda_t} \cdot \mathbf{K}^\top (\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right) \\
&= \frac{\exp \lambda_t}{1 + \exp \lambda_t} \cdot (\mathbf{K}\hat{\mathbf{c}}_t)^\top \cdot (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}_{t-1}) \,.
\end{aligned} \tag{3.36}
$$

Note how for the first iteration $t = 1$ in the context of denoising, where $\hat{\mathbf{x}}_0 = \mathbf{y}$ and $\mathbf{K} = \mathbf{I}$, the above will always be 0. In this case there is no difference between the previous estimate $\hat{\mathbf{x}}_{t-1}$ and the observation $\mathbf{y}$, so the likelihood part of Equation (2.23) does not matter for the energy and, as a result, the regularization weight λ_1 becomes meaningless.

Nonlinear Function Weights

Differentiating ℓ with respect to the function parameters $\boldsymbol{\omega}_{ti}$ yields

$$
\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\omega}_{ti}} &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \boldsymbol{\omega}_{ti}} \\
&= \hat{\mathbf{c}}_t^\top \cdot \left(-\mathbf{F}_{ti}^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \boldsymbol{\omega}_{ti}} \right) \\
&= -(\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \boldsymbol{\omega}_{ti}} \,,
\end{aligned} \tag{3.37}
$$

where the appropriate derivative from Section 3.1.3 can simply be plugged in.

Filter Weights

Recall from Section 3.1.3 that the filters represented by the matrices $\mathbf{F}_{ti}$ are built from a filter basis $\mathbf{B}$ with weight vectors $\tilde{\mathbf{f}}_{ti}$ as $\mathbf{f}_{ti} = \mathbf{B} \cdot \tilde{\mathbf{f}}_{ti}$.

However as [40] points out, learning both the function weights $\boldsymbol{\omega}_{ti}$ and the filter weights $\tilde{\mathbf{f}}_{ti}$ at arbitrary scale can lead to problems during training, because the parameters have too many degrees of freedom. The proposed solution is to constrain the filters to always have unit norm:

$$\mathbf{f} = \frac{\mathbf{B}\tilde{\mathbf{f}}}{\|\mathbf{B}\tilde{\mathbf{f}}\|} = \mathbf{B}\tilde{\mathbf{f}} \cdot \left(\tilde{\mathbf{f}}^\top \mathbf{B}^\top \mathbf{B}\tilde{\mathbf{f}}\right)^{-\frac{1}{2}}$$

For differentiating ℓ with respect to the filter weights $\tilde{\mathbf{f}}_{ti}$ under this constraint, the technique applied in [40] is to introduce a clique matrix $[x]_\mathcal{C}$ with $\mathbf{F}\mathbf{x} = (\mathbf{f}^\top [x]_\mathcal{C})^\top = [x]_\mathcal{C}^\top \mathbf{f}$.

Note that $[x]_\mathcal{C}$ is straightforward to construct from the image $\mathbf{x}$, as its columns are exactly the vectors $\mathbf{x}_{(c)}$ corresponding to the maximal cliques $c \in \mathcal{C}$ of the image, so that

$$[x]_\mathcal{C}^\top \mathbf{f} = \begin{bmatrix} \mathbf{x}_{(c_1)}^\top \\ \vdots \\ \mathbf{x}_{(c_\text{end})}^\top \end{bmatrix} \cdot \mathbf{f} = \begin{bmatrix} \mathbf{x}_{(c_1)}^\top \mathbf{f} \\ \vdots \\ \mathbf{x}_{(c_\text{end})}^\top \mathbf{f} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^\top \mathbf{x}_{(c_1)} \\ \vdots \\ \mathbf{f}^\top \mathbf{x}_{(c_\text{end})} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{c_1}^\top \mathbf{x} \\ \vdots \\ \mathbf{f}_{c_\text{end}}^\top \mathbf{x} \end{bmatrix} = \mathbf{F}\mathbf{x}\,.$$

Then, still following [40], the derivative of a term $\mathbf{F}\mathbf{x}$ with respect to the weights $\tilde{\mathbf{f}}$ becomes

$$\begin{aligned} \frac{\partial \mathbf{F}\mathbf{x}}{\partial \tilde{\mathbf{f}}} &= \frac{\partial \, [x]_\mathcal{C}^\top \mathbf{B}\tilde{\mathbf{f}} \cdot \left(\tilde{\mathbf{f}}^\top (\mathbf{B}^\top \mathbf{B})\tilde{\mathbf{f}}\right)^{-\frac{1}{2}}}{\partial \tilde{\mathbf{f}}} \\ &= \left([x]_\mathcal{C}^\top - \mathbf{F}\mathbf{x} \cdot \mathbf{f}^\top\right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}\|}\,, \end{aligned} \tag{3.38}$$

which can be used in

$$\begin{aligned} \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_\text{gt})}{\partial \tilde{\mathbf{f}}_{ti}} &= \hat{\mathbf{c}}_t^\top \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \hat{\mathbf{c}}_t^\top \cdot \left(-\frac{\partial \mathbf{F}_{ti}^\top \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \tilde{\mathbf{f}}_{ti}} \right) \\ &= -\frac{\partial (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= -\left((\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \tilde{\mathbf{f}}_{ti}} + \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^\top \cdot \frac{\partial (\mathbf{F}_{ti}\hat{\mathbf{c}}_t)}{\partial \tilde{\mathbf{f}}_{ti}} \right) \\ &= -\left((\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}} + \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^\top \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{c}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \right) \\ &= -\left((\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \operatorname{diag}\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}} + \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^\top \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{c}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \right) \end{aligned}$$

to obtain

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} = -\left((\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \operatorname{diag}\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\mathbf{x}}_{t-1}]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1} \cdot \mathbf{f}_{ti}^\top\right) + \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^\top \cdot \left([\hat{\mathbf{c}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{c}}_t \cdot \mathbf{f}_{ti}^\top\right)\right) \cdot \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|}. \tag{3.39}$$

3.3.7 Summary

With all of these gradients specified, the parameters of a truncated optimization model using gradient descent inference can be trained with respect to the loss over the training set, according to the bi-level framework explained in Section 3.1. A compact overview of the resulting equations for all quantities that need to be computed during training is provided in Figure 3.10.

It is important to note that neither the filter matrices $\mathbf{F}_{ti}$ nor the blur matrix $\mathbf{K}$ need to be explicitly constructed for the computations. Both only ever appear in conjunction with an image vector multiplied from the right, as in $\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}$ or $\mathbf{K}\hat{\mathbf{c}}_t$. The result of such an expression can be computed efficiently via a convolution of the image with the filter or blur kernel the matrix was based on, as in $\mathbf{f}_{ti} * \hat{\mathbf{x}}_{t-1}$ or $\mathbf{k} * \hat{\mathbf{c}}_t$. Multiplication of the transpose of a matrix with an image vector, such as $\mathbf{K}^\top(\mathbf{K}\hat{\mathbf{c}}_t)$, is equivalent to convolution of the image with the kernel rotated by 180 degrees and correct boundary handling [40, 13]. In both cases, the equality only holds for the central part of the image and artifacts are introduced at the image borders. To address these, in practice images are padded before applying the model and cropped before evaluating the loss, which is addressed in Section 4.1.2.

The results of a number of experiments with gradient descent models trained in this fashion are reported in Sections 4.2.1 and 4.3.1.

3.4 Half-Quadratic Inference

The second inference method to be considered here is a *multiplicative* variant of *half-quadratic inference* first explored in [11]. Its combination with the *Field of Experts* model has not been examined before in this truncated optimization framework.

In contrast to the gradient descent approach, the half-quadratic approach is not based on minimizing the energy function from Equation (2.23) directly, but rather seeks to exploit the advantages that come with a quadratic, Gaussian model by solving quadratic approximations of the actual model.

3.4.1 Motivation

Section 2.3.4 has shown how quadratic penalty functions ρ in the *Field of Experts* prior lead to Gaussian potential functions and thus to a Gaussian posterior distribution. The *maximum a-posteriori* solution in this case was simply the mean of the distribution and could be computed directly, without the need for an iterative method.

On the other hand, Section 2.3.3 explained how quadratic functions are a bad fit for modelling natural image statistics. What's more, the *radial basis function mixtures* used here, which were introduced in Section 3.1.3, are fully flexible and certainly not quadratic.

In order to still take advantage of the closed-form solution for Gaussian models, it is possible to formulate a non-Gaussian model in such a way that it becomes Gaussian when a set of specially introduced auxiliary variables is held fixed. This strategy is known as *half-quadratic* augmentation, and was first established in [19]. The term "half-quadratic" stems from the fact that the augmented model is quadratic only under specific conditions and otherwise retains its robust properties. An alternative form is described in [20] and provides the basis for the *Cascade of Shrinkage Fields* model in [40].

The specific *half-quadratic* variant that is relevant here can be found in [11], where a *half-quadratic* approach was first used to do *maximum a-posteriori* estimation.

Without going into too much detail, the augmentation works as follows. Each penalty function $\rho_i(u)$, at the location of each filter response u, is replaced by a function $\phi_i(u, z)$ that is quadratic for fixed z and approximates $\rho_i(u)$ at u. Distinct auxiliary variables z_{ic} are therefore needed for each filter $\mathbf{f}_i$ and each image clique c to cover all occuring values of u. Together, these auxiliary variables form one large auxiliary vector $\mathbf{z} \in \mathbb{R}^{N \cdot |\mathcal{C}|}$. When replacing the penalty functions like that, the prior from Equation (2.15) becomes

$$p(\mathbf{x}, \mathbf{z}) \propto \prod_{c \in C} \prod_{i=1}^{N} \exp\bigl(-\phi_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}, z_{ic})\bigr) \,. \tag{3.40}$$

To eliminate the auxiliary variables and again arrive at a distribution over $\mathbf{x}$, the type of *half-quadratic* augmentation considered here, which has been called the *envelope*

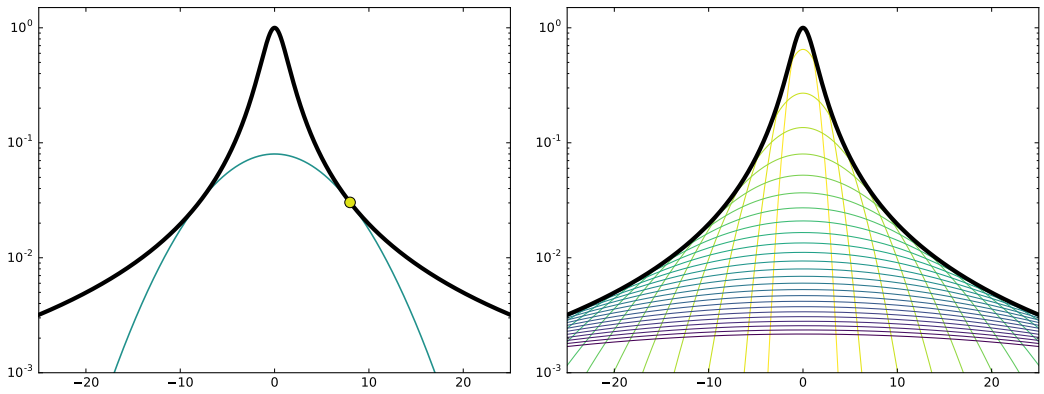


Figure 3.9: Half-quadratic approximation $\exp\big(-\phi(u,z)\big)$ of the robust Lorentzian potential $\exp\big(-\rho(u)\big) = \exp\big(-\log\left(1+\frac{1}{2}x^2\right)\big)$, as seen in Figure 2.8, for a specific value of z that is determined by u *(left)* and the tight lower bound induced by multiple such augmented potentials *(right)*

type, takes the maximum of Equation (3.40) over all $\mathbf{z}$,

$$\begin{aligned} p(\mathbf{x}) &\propto \max_{\mathbf{z}} p(\mathbf{x},\mathbf{z}) \\ &\propto \max_{\mathbf{z}} \prod_{c\in C}\prod_{i=1}^{N} \exp\left(-\phi_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}, z_{ic})\right), \end{aligned} \tag{3.41}$$

which also means that

$$\exp\left(-\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)})\right) \propto \max_{z_{ic}} \exp\left(-\phi_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}, z_{ic})\right) \tag{3.42}$$

and

$$\rho_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}) \propto \min_{z_{ic}} \phi_i(\mathbf{f}_i^\top \mathbf{x}_{(c)}, z_{ic})\,. \tag{3.43}$$

Following the *multiplicative* form of *half-quadratic* augmentation introduced in [19], the augmented penalty functions are of the general form

$$\phi(u,z) = \frac{1}{2}u^2 z + \psi(z)\,, \tag{3.44}$$

where the term $\psi(z)$ is added to ensure that Equation (3.42) holds. Its exact form depends on the penalty function $\rho(u)$ that is being approximated. Details on this can be found in [19] and [6] among others.

Plugging Equation (3.44) into Equation (3.43) then yields a minimization task which can be solved to find the value of z_{ic} that will be used to approximate ρ_i for the filter-clique combination $\mathbf{f}_i^\top \mathbf{x}_{(c)}$.

Essentially, when the model is specified in this way, the augmented potential functions $\exp\big(-\phi_i(u,\mathbf{z}_i)\big)$ combine to form a tight lower bound on each original potential function $\exp\big(-\rho_i(u)\big)$, as illustrated in Figure 3.9 for a common robust potential. The resulting probability $p(\mathbf{x})$, as the product of all potentials, can therefore never be overestimated be the augmented model, and maximizing the tight lower bound at the same time also maximizes this probability.

3.4.2 Update Formula

According to [11], *maximum a-posteriori* inference to find the pair

$$(\hat{\mathbf{x}}, \hat{\mathbf{z}}) = \arg\max_{\mathbf{x},\mathbf{z}} p(\mathbf{x}, \mathbf{z})$$

can be carried out by alternating between the steps of finding the image $\hat{\mathbf{x}}$ that maximizes $p(\mathbf{x} \mid \mathbf{y}, \hat{\mathbf{z}})$ and finding auxiliary variables $\hat{\mathbf{z}}$ that maximize $p(\mathbf{z} \mid \hat{\mathbf{x}}, \mathbf{y})$.

The respective update formulae for $\hat{\mathbf{z}}$ and $\hat{\mathbf{x}}$ can be stated as

$$\hat{z}_{t,ic} = \frac{\rho_i'(\mathbf{f}_i^\top \hat{\mathbf{x}}_{(c)t-1})}{\mathbf{f}_i^\top \hat{\mathbf{x}}_{(c)t-1}} \tag{3.45}$$

and

$$\hat{\mathbf{x}}_t = \left(\frac{\mathbf{K}^\top \mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_i^\top \cdot \operatorname{diag}\{\hat{z}_{t,ic} \mid c \in \mathcal{C}\} \cdot \mathbf{F}_i \right)^{-1} \cdot \left(\frac{\mathbf{K}^\top \mathbf{y}}{\sigma_t^2} \right), \tag{3.46}$$

the latter of which should be familiar from the Gaussian random field model in Section 2.3.4. Together, they form one iteration of *half-quadratic inference* in the sense of this thesis.

The filter responses $\mathbf{f}_i^\top \hat{\mathbf{x}}_{(c)t-1}$ in Equation (3.45) can again be written more concisely with the filter matrix formulation from Section 3.3.2, yielding an auxiliary vector

$$\hat{\mathbf{z}}_{t,i} = \frac{\rho_i'(\mathbf{F}_i \hat{\mathbf{x}}_{t-1})}{\mathbf{F}_i \hat{\mathbf{x}}_{t-1}}\,. \tag{3.47}$$

In addition, the penalty functions ρ_i again only appear in the form of their derivatives ρ_i', so the derivatives could once more be modelled directly as in Section 3.3.2. But in this case, the whole of Equation (3.47) is a function of the filter response vector $\mathbf{F}_i \hat{\mathbf{x}}_{t-1}$ and can thus be modelled directly by a nonlinear function as

$$\hat{\mathbf{z}}_{t,i} = \rho_i(\mathbf{F}_i \hat{\mathbf{x}}_{t-1}) = \frac{\rho_i^{*\prime}(\mathbf{F}_i \hat{\mathbf{x}}_{t-1})}{\mathbf{F}_i \hat{\mathbf{x}}_{t-1}} \tag{3.48}$$

with $\rho^*(u)$ denoting the actual penalty function.

With these changes, the consolidated update formula for one iteration of *half-quadratic inference* takes the form

$$\begin{aligned} \hat{\mathbf{x}}_t &= \left(\frac{\mathbf{K}^\top \mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right)^{-1} \cdot \left(\frac{\mathbf{K}^\top \mathbf{y}}{\sigma_t^2} \right) \\ &= \boldsymbol{\Omega}_t^{-1} \cdot \boldsymbol{\eta}_t \end{aligned} \tag{3.49}$$

with

$$\boldsymbol{\Omega}_t = \frac{\mathbf{K}^\top \mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti} \hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \quad \text{and}$$

$$\boldsymbol{\eta}_t = \frac{\mathbf{K}^\top \mathbf{y}}{\sigma_t^2}\,.$$

As before in the truncated optimization scheme, each step t uses a separate set of parameters. In addition to the linear filter weights $\tilde{\mathbf{f}}_{ti}$ and nonlinear function weights $\boldsymbol{\omega}_{ti}$, the *half-quadratic* inference steps have a parameter σ_t that is connected to the noise level present in the processed images and will be learned as well.

Note that the update formula stated above can be transformed by introducing a redundant term $(\sigma_t^2)^{-1} \cdot \sigma_t^2 = 1$ to obtain

$$\begin{aligned}\hat{\mathbf{x}}_t &= \boldsymbol{\Omega}_t^{-1} \cdot (\sigma_t^2)^{-1} \cdot \sigma_t^2 \cdot \boldsymbol{\eta}_t \\ &= \left(\mathbf{K}^\top \mathbf{K} + \sigma_t^2 \cdot \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right)^{-1} \cdot \mathbf{K}^\top \mathbf{y}\,,\end{aligned}$$

where it becomes clear that σ_t^2 takes the place of a *regularization weight* in the update formula, and could itself be subsumed in the scaling of the nonlinear functions ρ_{ti}. In practice however, it would require the simultaneous scaling of every single weight parameter in each function to mimic the effect of σ_t^2, which may take a long time to achieve for the training algorithm. The *noise level parameter* σ_t can thus be interpreted as an additional shared scaling factor affecting all nonlinear functions, possibly making training easier. Whether that is the case in practice was not investigated in this thesis.

3.4.3 Solving Systems of Linear Equations

For each step of half-quadratic inference, it is necessary to calculate $\hat{\mathbf{x}}_t = \boldsymbol{\Omega}_t^{-1} \cdot \boldsymbol{\eta}_t$. While the previous section showed a straightforward way to construct $\boldsymbol{\Omega}_t \in \mathbb{R}^{mn \times mn}$, the computational complexity of finding its inverse $\boldsymbol{\Omega}_t^{-1}$ is cubic[1] in the number $m \cdot n$ of pixels in the image – even for everyday photographs, inverting $\boldsymbol{\Omega}_t$ quickly becomes prohibitive.

To alleviate this problem, $\boldsymbol{\Omega}_t$ can be multiplied from the left on both sides to get $\boldsymbol{\Omega}_t \hat{\mathbf{x}}_t = \boldsymbol{\eta}_t$, which is a system of linear equations. Such systems can often be solved more efficiently by finding a suitable decomposition of the system matrix $\boldsymbol{\Omega}_t$ into two triangular matrices. Then only two very simple systems of linear equations need to be solved instead of a single more difficult one.

The usual decomposition of choice, both for speed and for numerical stability, is the *Cholesky decomposition* [16]. It produces a lower-triangular matrix $\mathbf{L}$ such that

$$\boldsymbol{\Omega}_t = \mathbf{L}\mathbf{L}^\top\,.$$

Solving the system $\boldsymbol{\Omega}_t \hat{\mathbf{x}}_t = \mathbf{L}\mathbf{L}^\top \hat{\mathbf{x}}_t = \boldsymbol{\eta}_t$ is then equivalent to first solving

$$\mathbf{L}\boldsymbol{\mu} = \boldsymbol{\eta}_t$$

for $\boldsymbol{\mu}$, and then

$$\mathbf{L}^\top \hat{\mathbf{x}}_t = \boldsymbol{\mu}$$

[1] While this is true for dense matrices, it does not generally hold for sparse matrices. Determining the complexity of algorithms on sparse matrices, however, is often not trivial, and the details of this problem are skipped in this thesis.

for $\hat{\mathbf{x}}_t$, both of which are easy to compute via forward and back substitution, respectively. The same factorization $\mathbf{L}\mathbf{L}^\top$ can then be used to solve further equation systems characterized by $\boldsymbol{\Omega}_t$, one of which will come up during parameter learning.

It is important to note that although this decomposition grants an improvement in computation time of factor 2 to 3 over matrix inversion and some other direct methods, it does not solve the general issue of cubic computational complexity. Its speedup is nonetheless of great help, and worth a bit of extra effort.

Namely, in order to use the Cholesky decomposition, it is necessary to first make sure that $\boldsymbol{\Omega}_t$ is both *symmetric* and *positive-definite*. The former, which means nothing else than $\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_t^\top$, is easy to see:

$$\begin{aligned}
\boldsymbol{\Omega}_t^\top &= \left(\frac{\mathbf{K}^\top \mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right)^\top \\
&= \frac{\left(\mathbf{K}^\top \mathbf{K}\right)^\top}{\sigma_t^2} + \sum_{i=1}^{N} \left(\mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right)^\top \\
&= \frac{\mathbf{K}^\top \mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \\
&= \boldsymbol{\Omega}_t \,. \qquad (3.50)
\end{aligned}$$

For a matrix $\mathbf{M}$ to be called *positive-definite*, it has to hold that

$$\mathbf{x}^\top \cdot \mathbf{M} \cdot \mathbf{x} > 0 \quad \text{for any } \mathbf{x} \neq \vec{\mathbf{0}} \,. \qquad (3.51)$$

If the product can also be equal to zero, $\mathbf{M}$ is called *positive-semidefinite* instead. The vector $\mathbf{x}$ in this definition is not related to the images discussed in the rest of this thesis.

To ensure *positive-definiteness* of the matrix $\boldsymbol{\Omega}_t$, the values of the nonlinear functions ρ_{ti} need to be constrained to be strictly positive, as explained in Section 3.1.3. Then it is possible to write

$$\operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} = \mathbf{D}^\top \mathbf{D} \quad \text{with} \quad \mathbf{D} = \operatorname{diag}\Big\{ \sqrt{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})} \Big\} \,, \qquad (3.52)$$

which helps decompose the term inside the sum from Equation (3.49) to find

$$\begin{aligned}
\mathbf{x}^\top \cdot (\mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}) \cdot \mathbf{x} &= \mathbf{x}^\top \cdot \mathbf{F}_{ti}^\top \cdot \mathbf{D}^\top \mathbf{D} \cdot \mathbf{F}_{ti} \cdot \mathbf{x} \\
&= (\mathbf{D}\mathbf{F}_{ti}\mathbf{x})^\top \cdot (\mathbf{D}\mathbf{F}_{ti}\mathbf{x}) \\
&= \|\mathbf{D}\mathbf{F}_{ti}\mathbf{x}\|^2 \\
&\geq 0 \quad \text{for any } \mathbf{x} \neq \vec{\mathbf{0}} \,, \qquad (3.53)
\end{aligned}$$

meaning that each matrix $\mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}$ is *positive-semidefinite*.

Recall from Section 2.2.3 that the blur matrix $\mathbf{K}$ represents the effect of a *point spread function*, so it can not have negative entries and not all of its entries can be

zero. As a result,

$$\begin{aligned}\mathbf{x}^\top \cdot \frac{\mathbf{K}^\top\mathbf{K}}{\sigma_t^2} \cdot \mathbf{x} &= \frac{1}{\sigma_t^2} \cdot \mathbf{x}^\top \cdot \mathbf{K}^\top\mathbf{K} \cdot \mathbf{x} \\ &= \frac{1}{\sigma_t^2} \cdot (\mathbf{K}\mathbf{x})^\top \cdot (\mathbf{K}\mathbf{x}) \\ &= \frac{1}{\sigma_t^2} \cdot \|\mathbf{K}\mathbf{x}\|^2 \\ &> 0 \quad \text{for any } \mathbf{x} \neq \vec{\mathbf{0}} \end{aligned} \tag{3.54}$$

shows that $(\mathbf{K}^\top\mathbf{K})/\sigma_t^2$ is a *positive-definite* matrix.

Putting it all together, it holds that

$$\begin{aligned}\mathbf{x}^\top \cdot \boldsymbol{\Omega}_t \cdot \mathbf{x} &= \mathbf{x}^\top \cdot \left(\frac{\mathbf{K}^\top\mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right) \cdot \mathbf{x} \\ &= \mathbf{x}^\top \cdot \frac{\mathbf{K}^\top\mathbf{K}}{\sigma_t^2} \cdot \mathbf{x} + \sum_{i=1}^{N} \mathbf{x}^\top \cdot (\mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}) \cdot \mathbf{x} \\ &= \frac{1}{\sigma_t^2} \cdot \|\mathbf{K}\mathbf{x}\|^2 + \sum_{i=1}^{N} \|\mathbf{D}\mathbf{F}_{ti}\mathbf{x}\|^2 \\ &> 0 \quad \text{for any } \mathbf{x} \neq \vec{\mathbf{0}}\,. \end{aligned} \tag{3.55}$$

So $\boldsymbol{\Omega}_t$, being the sum of a *positive-definite* matrix and N *positive-semidefinite* matrices, is itself again *positive-definite* and the Cholesky decomposition can be employed to solve $\boldsymbol{\Omega}_t\hat{\mathbf{x}}_t = \boldsymbol{\eta}_t$ for $\hat{\mathbf{x}}_t$.

3.4.4 Derivative of the Loss Function

In the next few sections, the gradients needed for parameter training as explained in Section 3.2 will be derived. Since the multiplicative half-quadratic approach has not yet been examined in a truncated optimization setting, the derivations that follow are new contributions to the field of image restoration.

The first step, like in the gradient descent case, is to differentiate the overall loss $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})$ with respect to all model parameters of a specific step t. For this purpose, it helps to first find a more general derivative and proceed from there with the individual parameters.

The derivative of $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})$ with respect to any variable v_t occurring in $\hat{\mathbf{x}}_t$ from Equation (3.49) will generally be of the form

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial v_t} &= \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \left(\boldsymbol{\Omega}_t^{-1} \cdot \boldsymbol{\eta}_t \right)}{\partial v_t} \\ &= \hat{\mathbf{c}}_t^\top \cdot \left(\boldsymbol{\Omega}_t^{-1} \frac{\partial \boldsymbol{\eta}_t}{\partial v_t} + \frac{\partial \boldsymbol{\Omega}_t^{-1}}{\partial v_t} \boldsymbol{\eta}_t \right) \\ &= \hat{\mathbf{c}}_t^\top \cdot \left(\boldsymbol{\Omega}_t^{-1} \frac{\partial \boldsymbol{\eta}_t}{\partial v_t} - \boldsymbol{\Omega}_t^{-1} \cdot \frac{\partial \boldsymbol{\Omega}_t}{\partial v_t} \cdot \boldsymbol{\Omega}_t^{-1} \cdot \boldsymbol{\eta}_t \right) \end{aligned}$$

$$= -\left(\hat{\mathbf{c}}_t^\top \cdot \boldsymbol{\Omega}_t^{-1}\right) \cdot \left(\frac{\partial \boldsymbol{\Omega}_t}{\partial v_t} \cdot \left(\boldsymbol{\Omega}_t^{-1} \cdot \boldsymbol{\eta}_t\right) - \frac{\partial \boldsymbol{\eta}_t}{\partial v_t}\right)$$
$$= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(\frac{\partial \boldsymbol{\Omega}_t}{\partial v_t} \cdot \hat{\mathbf{x}}_t - \frac{\partial \boldsymbol{\eta}_t}{\partial v_t}\right) \tag{3.56}$$

with vector

$$\hat{\boldsymbol{\varsigma}}_t = -\left(\hat{\mathbf{c}}_t^\top \cdot \boldsymbol{\Omega}_t^{-1}\right)^\top = -\boldsymbol{\Omega}_t^{-1} \cdot \hat{\mathbf{c}}_t\,, \tag{3.57}$$

where the last transformation is possible because $\boldsymbol{\Omega}_t$ is a symmetric matrix (*cf.* Equation (3.50)) and thus it holds that $(\boldsymbol{\Omega}_t^{-1})^\top = \boldsymbol{\Omega}_t^{-1}$.

The chain rule term

$$\hat{\mathbf{c}}_t^\top = \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_t} \tag{3.58}$$

plays the same role it had in the gradient descent case (*cf.* Equation (3.34)).

Since $\hat{\mathbf{x}}_t$ is already computed in the forward pass of the backpropagation algorithm and can be reused when computing the gradients, and the same factorization of $\boldsymbol{\Omega}_t$ used to find $\hat{\mathbf{x}}_t$ can be reused to compute $\hat{\boldsymbol{\varsigma}}_t = -(\boldsymbol{\Omega}_t^{-1} \cdot \hat{\mathbf{c}}_t)$. All that is left to do is to find the derivatives of $\boldsymbol{\Omega}_t$ and $\boldsymbol{\eta}_t$ with respect to the noise level parameter σ_t, the weights defining each ρ_{ti} and $\mathbf{F}_{ti}$, as well as the previous estimate $\hat{\mathbf{x}}_{t-1}$.

3.4.5 Input Gradient

To pass a chain rule term back to the previous step for backpropagation, it is once more necessary to find the derivative of the loss function with respect to the current step's input $\hat{\mathbf{x}}_{t-1}$. At first glance, this involves the derivative of a matrix with respect to a vector, which would yield a three-dimensional object and make the whole derivation unnecessarily difficult.

Fortunately, this problem can be sidestepped with the use of the element-wise operator introduced in Section 2.1:

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_{t-1}} = \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial \boldsymbol{\Omega}_t}{\partial \hat{\mathbf{x}}_{t-1}} \cdot \hat{\mathbf{x}}_t$$
$$= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}\right) \cdot \hat{\mathbf{x}}_t$$
$$= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti}\right)\right) \cdot \hat{\mathbf{x}}_t$$

At this point, $\hat{\boldsymbol{\varsigma}}_t^\top$ and $\hat{\mathbf{x}}_t$ can be drawn into the sum. Note however that they will not be differentiated with respect to $\hat{\mathbf{x}}_{t-1}$ as they are not part of the actual differential. It is best to think of them as arbitrary vectors for the purpose of this derivation.

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_{t-1}} = \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti}\right) \cdot \hat{\mathbf{x}}_t\,.$$

Equations (2.2) and (2.3) now allow the transformations

$$
\begin{aligned}
&= \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \big(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot (\mathbf{F}_{ti}\hat{\mathbf{x}}_t)\big) \\
&= \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{t-1}} \big(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot (\mathbf{F}_{ti}\hat{\mathbf{x}}_t)\big) \\
&= \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \hat{\mathbf{x}}_{t-1}}\right) \\
&= \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \left(\frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \hat{\mathbf{x}}_{t-1}}\right)\right) \\
&= \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \Big((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \big(\mathrm{diag}\big\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\big\} \cdot \mathbf{F}_{ti}\big)\Big) \\
&= \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \big((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti}\big)
\end{aligned}
\tag{3.59}
$$

to arrive at a solution for the input gradient of the t^{th} step.

The result of this can now be passed back as an updated chain rule term $\hat{\mathbf{c}}_{t-1}^\top$ to the previous step, to assist in computing that step's parameter gradients as explained in Section 3.2.1.

3.4.6 Parameter Gradients

The following passages show how the loss function $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})$ can be differentiated with respect to the model parameters σ_t, $\boldsymbol{\omega}_{ti}$ and $\tilde{\mathbf{f}}_{ti}$ of the t^{th} step.

Noise level

The derivative of Equation (3.49) with respect to σ_t is straightforward to obtain from Equation (3.56) as

$$
\begin{aligned}
\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \sigma_t} &= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(\frac{\partial \boldsymbol{\Omega}_t}{\partial \sigma_t} \cdot \hat{\mathbf{x}}_t - \frac{\partial \boldsymbol{\eta}_t}{\partial \sigma_t}\right) \\
&= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(\frac{\partial \mathbf{K}^\top\mathbf{K}/\sigma_t^2}{\partial \sigma_t} \cdot \hat{\mathbf{x}}_t - \frac{\partial \mathbf{K}^\top\mathbf{y}/\sigma_t^2}{\partial \sigma_t}\right) \\
&= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(-2\frac{1}{\sigma_t^3}\mathbf{K}^\top\mathbf{K} \cdot \hat{\mathbf{x}}_t + 2\frac{1}{\sigma_t^3}\mathbf{K}^\top\mathbf{y}\right) \\
&= -2\frac{1}{\sigma_t^3} \cdot \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(\mathbf{K}^\top\mathbf{K} \cdot \hat{\mathbf{x}}_t - \mathbf{K}^\top\mathbf{y}\right) \\
&= 2\frac{1}{\sigma_t^3} \cdot (\mathbf{K}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}_t) \ .
\end{aligned}
\tag{3.60}
$$

An alternative option here would be to replace $1/\sigma_t^2$ in Equation (3.49) with a positive-constrained term $\sigma_t^+ = \ln\left(1 + \exp\left(\sigma_t\right)\right)$, like it was done with the regularization weight λ_t^+ in Section 3.3.3. This would result in a gradient

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \sigma_t} = \frac{\exp\left(\sigma t\right)}{1 + \exp\left(\sigma t\right)} \cdot \left(\mathbf{K}\hat{\boldsymbol{\varsigma}}_t\right)^\top \cdot \left(\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}_t\right)$$

instead, which does not contain σ_t in a cubic form and can practically not become zero. But since Equation (3.60) caused no problems during the experiments conducted for this thesis and the positivity constraint may introduce new difficulties as described in Section 3.1.3, this option was not investigated further.

Nonlinear Function Weights

In a way that is very similar to the derivation of the input gradient in Section 3.4.5, the loss $\ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})$ can be differentiated with respect to the weights $\boldsymbol{\omega}_{ti}$ of function ρ_{ti} as follows:

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \boldsymbol{\omega}_{ti}} &= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial \boldsymbol{\Omega}_t}{\partial \boldsymbol{\omega}_{ti}} \cdot \hat{\mathbf{x}}_t \\ &= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial}{\partial \boldsymbol{\omega}_{ti}} \left(\mathbf{F}_{ti}^\top \cdot \mathrm{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}\right) \cdot \hat{\mathbf{x}}_t\end{aligned}$$

Here $\hat{\boldsymbol{\varsigma}}_t^\top$ and $\hat{\mathbf{x}}_t$ can be drawn in without any trouble as they do not depend on $\boldsymbol{\omega}_{ti}$:

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \boldsymbol{\omega}_{ti}} = \frac{\partial}{\partial \boldsymbol{\omega}_{ti}} \left(\hat{\boldsymbol{\varsigma}}_t^\top \cdot \mathbf{F}_{ti}^\top \cdot \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti}\right) \cdot \hat{\mathbf{x}}_t\right)$$

Then the associativity from Equation (2.3) and the product rule from Equation (2.2) can be used to obtain

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \boldsymbol{\omega}_{ti}} &= \frac{\partial}{\partial \boldsymbol{\omega}_{ti}} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot (\mathbf{F}_{ti}\hat{\mathbf{x}}_t)\right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \frac{\partial}{\partial \boldsymbol{\omega}_{ti}} \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot (\mathbf{F}_{ti}\hat{\mathbf{x}}_t)\right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \boldsymbol{\omega}_{ti}}\right) . \end{aligned} \tag{3.61}$$

Just as in the gradient descent case, it only remains to plug in the appropriate derivative for ρ_{ti} from Section 3.1.3 to complete the gradient.

Filter Weights

For differentiating the update formula with relation to the filter weights $\tilde{\mathbf{f}}_{ti}$ that define the filter matrix $\mathbf{F}_{ti}$, the general from from Equation (3.56) can be expanded

as

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} &= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \frac{\partial \boldsymbol{\Omega}_t}{\partial \tilde{\mathbf{f}}_{ti}} \cdot \hat{\mathbf{x}}_t \\ &= \hat{\boldsymbol{\varsigma}}_t^\top \cdot \left(\frac{\partial \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}}{\partial \tilde{\mathbf{f}}_{ti}} \right) \cdot \hat{\mathbf{x}}_t \\ &= \frac{\partial (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot (\mathbf{F}_{ti}\hat{\mathbf{x}}_t)}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \frac{\partial \mathbf{u}^\top \cdot \operatorname{diag}\{\mathbf{v}\} \cdot \mathbf{w}}{\partial \tilde{\mathbf{f}}_{ti}}\end{aligned}$$

with the three vectors

$$\mathbf{u} = \mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t\,, \qquad \mathbf{v} = \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\,, \qquad \mathbf{w} = \mathbf{F}_{ti}\hat{\mathbf{x}}_t\,. \tag{3.62}$$

This can be rewritten using the element-wise operator $\odot$ and exploiting the product rule from Equation (2.2) to get

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} &= \frac{\partial \mathbf{u}^\top \cdot (\mathbf{v} \odot \mathbf{w})}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \mathbf{u}^\top \frac{\partial (\mathbf{v} \odot \mathbf{w})}{\partial \tilde{\mathbf{f}}_{ti}} + (\mathbf{v} \odot \mathbf{w})^\top \frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \mathbf{u}^\top \left(\mathbf{v} \odot \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{f}}_{ti}} + \mathbf{w} \odot \frac{\partial \mathbf{v}}{\partial \tilde{\mathbf{f}}_{ti}} \right) + (\mathbf{v} \odot \mathbf{w})^\top \frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \mathbf{u}^\top \left(\mathbf{v} \odot \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{f}}_{ti}} \right) + \mathbf{u}^\top \left(\mathbf{w} \odot \frac{\partial \mathbf{v}}{\partial \tilde{\mathbf{f}}_{ti}} \right) + (\mathbf{v} \odot \mathbf{w})^\top \frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{f}}_{ti}}\,,\end{aligned} \tag{3.63}$$

where for the first part, it simply is

$$\begin{aligned}\mathbf{u}^\top \left(\mathbf{v} \odot \frac{\partial \mathbf{w}}{\partial \tilde{\mathbf{f}}_{ti}} \right) &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_t}{\partial \tilde{\mathbf{f}}_{ti}}\,.\end{aligned} \tag{3.64}$$

For the second part, it is

$$\begin{aligned}\mathbf{u}^\top \left(\mathbf{w} \odot \frac{\partial \mathbf{v}}{\partial \tilde{\mathbf{f}}_{ti}} \right) &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \tilde{\mathbf{f}}_{ti}} \right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \left(\frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}} \right) \right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \left(\operatorname{diag}\big\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\big\} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}} \right) \right) \\ &= (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \frac{\partial \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}}{\partial \tilde{\mathbf{f}}_{ti}} \right)\,.\end{aligned} \tag{3.65}$$

And, finally, for the third part, it is

$$\begin{aligned}(\mathbf{v} \odot \mathbf{w})^\top \frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{f}}_{ti}} &= \left(\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti}\hat{\mathbf{x}}_t\right)^\top \cdot \frac{\partial \mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \left(\operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}\hat{\mathbf{x}}_t\right)^\top \cdot \frac{\partial \mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \\ &= \left(\mathbf{F}_{ti}\hat{\mathbf{x}}_t\right)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \frac{\partial \mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t}{\partial \tilde{\mathbf{f}}_{ti}} \,. \end{aligned} \tag{3.66}$$

Like in the gradient descent case, the filters need to be learned in a normalized form, so that the scaling of the filters and nonlinear functions does not create ambiguity during training.

To this end, it is again possible to use Equation (3.38), taken from [40], to replace each instance of $\partial \mathbf{F}_{ti}(\cdot)/\partial \tilde{\mathbf{f}}_{ti}$ in the above expressions and obtain the final formula for the filter weights gradient as

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} = {} & \left(\mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t\right)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\mathbf{x}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_t \cdot \mathbf{f}_{ti}^\top\right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} + \\ & \left(\mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t\right)^\top \Bigg(\left(\mathbf{F}_{ti}\hat{\mathbf{x}}_t\right) \odot \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \\ & \quad \left(\left([\hat{\mathbf{x}}_{t-1}]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1} \cdot \mathbf{f}_{ti}^\top\right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} \right) \Bigg) + \\ & \left(\mathbf{F}_{ti}\hat{\mathbf{x}}_t\right)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\boldsymbol{\zeta}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\boldsymbol{\zeta}}_t \cdot \mathbf{f}_{ti}^\top\right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} \,. \end{aligned} \tag{3.67}$$

While this turns out to be a relatively complex calculation, in practice many parts can be precomputed or reused, as they are the same for each training image or are shared with other equations from this section.

3.4.7 Summary

This concludes the derivations for the training of a half-quadratic model under the truncated bi-level framework that is the context for this thesis. In comparison to the gradient descent approach, it is already clear that the computational demands of this variant of half-quadratic inference are much higher. Each iteration requires the solution of an equation system that can become very large for even moderate image sizes. Solving this system with a direct method, such as the Cholesky decomposition, makes it necessary to explicitly construct the filter matrices $\mathbf{F}_{ti}$ and the blur matrix $\mathbf{K}$ instead of short-cutting their application via convolution operations as described in Section 3.3.7. Construction and multiplication of these sparse matrices is both time and memory consuming, as the number of their non-zero entries grows with the size of the filters, blur kernels and processed images.

Experimental results for both types of model considered here are presented in the next chapter, followed by a discussion of their respective weak and strong points.

A compact overview of all equations needed for training the truncated half-quadratic model presented here is given in Figure 3.11. This overview omits the definitions of some additional variables for brevity, these definitions can be found in the appropriate sections above.

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) + \lambda_t^+ \cdot \mathbf{K}^\top(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y}) \right)$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_{t-1}} = \hat{\mathbf{c}}_t^\top - \left(\sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti}\hat{\mathbf{c}}_t + \lambda_t^+ \cdot \mathbf{K}^\top\mathbf{K}\hat{\mathbf{c}}_t \right)^\top$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \lambda_t} = \frac{\exp \lambda_t}{1 + \exp \lambda_t} \cdot (\mathbf{K}\hat{\mathbf{c}}_t)^\top \cdot (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}_{t-1})$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\omega}_{ti}} = -(\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \boldsymbol{\omega}_{ti}}$$

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} = &- \Big((\mathbf{F}_{ti}\hat{\mathbf{c}}_t)^\top \cdot \operatorname{diag}\{\rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\mathbf{x}}_{t-1}]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1} \cdot \mathbf{f}_{ti}^\top \right) + \\ &\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})^\top \cdot \left([\hat{\mathbf{c}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{c}}_t \cdot \mathbf{f}_{ti}^\top \right) \Big) \cdot \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|}\end{aligned}$$

Figure 3.10: Equations needed for training the truncated gradient descent model.

$$\hat{\mathbf{x}}_t = \left(\frac{\mathbf{K}^\top\mathbf{K}}{\sigma_t^2} + \sum_{i=1}^{N} \mathbf{F}_{ti}^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \mathbf{F}_{ti} \right)^{-1} \cdot \left(\frac{\mathbf{K}^\top\mathbf{y}}{\sigma_t^2} \right)$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \hat{\mathbf{x}}_{t-1}} = \sum_{i=1}^{N} (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \mathbf{F}_{ti} \right)$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \sigma_t} = 2\frac{1}{\sigma_t^3} \cdot (\mathbf{K}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}_t)$$

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \boldsymbol{\omega}_{ti}} = (\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \left((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \frac{\partial \rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})}{\partial \boldsymbol{\omega}_{ti}} \right)$$

$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\text{gt}})}{\partial \tilde{\mathbf{f}}_{ti}} = &(\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\mathbf{x}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_t \cdot \mathbf{f}_{ti}^\top \right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} + \\ &(\mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t)^\top \Bigg((\mathbf{F}_{ti}\hat{\mathbf{x}}_t) \odot \rho'_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1}) \odot \\ &\qquad \left(\left([\hat{\mathbf{x}}_{t-1}]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1} \cdot \mathbf{f}_{ti}^\top \right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|} \right) \Bigg) + \\ &(\mathbf{F}_{ti}\hat{\mathbf{x}}_t)^\top \cdot \operatorname{diag}\{\rho_{ti}(\mathbf{F}_{ti}\hat{\mathbf{x}}_{t-1})\} \cdot \left([\hat{\boldsymbol{\varsigma}}_t]_{\mathcal{C}}^\top - \mathbf{F}_{ti}\hat{\boldsymbol{\varsigma}}_t \cdot \mathbf{f}_{ti}^\top \right) \frac{\mathbf{B}}{\|\mathbf{B}\tilde{\mathbf{f}}_{ti}\|}\end{aligned}$$

Figure 3.11: Equations needed for training the truncated half-quadratic model.

4 Experiments and Results

The aim of this thesis is the implementation and comparison of the two truncated inference approaches presented in the previous chapter, for the applications of *image denoising* and *non-blind image deconvolution*. This chapter describes the setup for the different experiments that were carried out to this end, and reports their results in comparison to each other and to related work.

4.1 Training and Test Setup

For the implementation of the two models and their inference and training algorithms, *The Julia Language* [4, 3] was chosen, as it provides uncomplicated syntax similar to *MATLAB* or *Python* while promising *C*-level speed due to its sophisticated just-in-time compiler and type system. As the language was created with technical computing in mind, it comes with most mathematical functions already available and many more accessible through additional packages.

While implementing half-quadratic inference, a problem manifested itself in that sparse matrix operations, which constitute the computational backbone of the half-quadratic approach, are not particularly well optimized in native *Julia*. Thanks to *Julia*'s support for calling *Python* functions, however, it was possible to exploit the much faster sparse matrix multiplication provided by the *SciPy* library.

Two other bottlenecks encountered were the evaluation of the nonlinear functions and convolution with the linear filters, chiefly because these operations need to be carried out very many times during each iteration of the training procedure. Although the function evaluation is already sped up by the use of lookup tables, finding the interpolated lookup table value for each filter response on an image still proved to be time consuming. A further improvement is achieved with the direct use of parallelized *C* code, called from within *Julia*. Convolutions are sped up with the help of *Julia*'s powerful metaprogramming capabilities and optimized handling of the image borders. Both the *C* code and the fast convolution implementation were provided by Uwe Schmidt.

Yet even after these improvements, the overall runtime of the code used here remained significantly worse than that of other published approaches, in particular the *MATLAB* implementation of truncated gradient descent denoising by Chen and Pock [13]. Where they report a training time of about 21 hours for a model with five steps and 48 experts of size 7×7 when using 180×180 patches from all 400 training images, the code used here takes almost as much time to train a five step model with 8 3×3 experts on the same data set, using a similar machine. The main reason for this large difference may be found in their heavy use of parallelization over the training images with eight CPUs. Although parts of the implementation for this

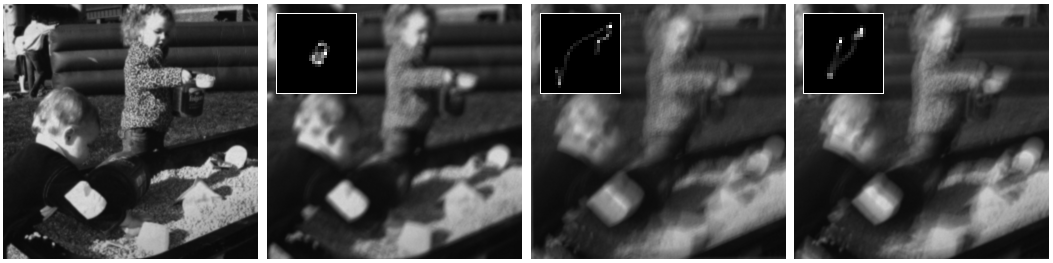


Figure 4.1: *(Left to right)* The first ground truth image in the deblurring test set, courtesy of [29], blurred with the third, fourth and eighth blur kernel of the same set respectively. The kernels shown here are scaled up by a factor of three for visibility.

thesis make use of parallelization, in general it lacks this kind of efficiency. A further speedup of the gradient descent approach is possible by moving computations to the GPU, which was not part of this thesis.

The half-quadratic inference variant employed by Schmidt and Roth in [40] differs from the one investigated here, in that although it requires the solution of large equation systems, these can be solved very fast with the help of *discrete Fourier transforms*. The difference in computational complexity means that the approach here could not be applied to the same size of input images or amount of training data, putting a limit on the results that could be achieved.

4.1.1 Data Sets

To enable a sensible comparison with the results of existing methods, the denoising experiments for this thesis were carried out with data from the same image set used in the majority of related work [35, 39, 2, 40, 13]. This data set is called the *Berkeley Segmentation Data Set*, or *BSDS500* [1], and is a collection of 500 coloured photographs of natural scenes divided into a training, a test and a validation portion. In [35], Roth and Black established a set of 68 images from the validation portion as a common denoising benchmark, which also constitute the test set for the denoising experiments reported here. These test images were used at full resulution, which is 321×481 pixels or vice versa depending on orientation. They were converted to grayscale, and an effort was made to apply the exact same noise as in the MATLAB implementations of other authors. A sample image from this set can be seen in Figure 3.2.

For the training set in the denoising experiments, a certain number of images from the training and test portions of *BSDS500* were taken and central patches of a fixed size extracted. Due to the technical limitations mentioned above, it was not feasible to exploit as many and as large patches as other, more efficient implementations managed to do [40, 13].

For deblurring, a set of 4 different grayscale images and 8 different blur kernels, yielding a total of 32 test image pairs, was introduced by Levin *et al.* in [29] and has been used by Schmidt *et al.* to compare results in [41] and [40]. Example images from this set can be seen in Figure 4.1. The same test set was used here to measure the performance of both approaches for deblurring. As in the denoising experiments, the images in the test set were used at their full resolution, which is 255×255

Figure 4.2: Some of the realistic blur kernels generated for the training set in [41] and used here for the same purpose.

pixels. However, the ground truths and observations in this data set do not perfectly align with the provided kernels, so the predicted deblurred images need to be shifted slightly to account for this. A function adapted from [29] was used here to find the shift which best aligns prediction and ground truth. This function also dismisses the outer 16 pixels of both images, but since an identical or very similar function was used in other works as well, it should not undermine the comparison.

Training was carried out on image patches from the *Berkeley Segmentation Data Set*, which were blurred with a subset of the realistic blur kernels generated by the authors of [41]. Each image was blurred with a different kernel. These kernels range in size from 15×15 to 37×37 pixels, but since the largest kernel in the test set is of size 27×27, larger kernels were cropped to this size and re-normalized for training in order to reduce computational cost. A few of the kernels used in this way are shown in Figure 4.2.

4.1.2 Boundary Handling

As mentioned in the previous section, the application of filters in each inference step leads to undesirable effects at the image border. The easiest solution to this problem is to pad input images with an artificial border region that is of no interest, and then crop the output to remove the then distorted and superfluous border before the loss is computed. This leaves the actual image unaffected, provided the padding was large enough with respect to the filter size.

In the case of deconvolution, an additional matter is the size difference between ground truth and observation mentioned in Section 2.2.3. Here, the observations need to be padded already to obtain initial guesses of sufficient size for use as input images. Since the reconstruction of this added border region from the observation and the known blur kernel is usually poor in the first place and will be discarded before loss computation, it can simultaneously fill the role of the artificial border region described above. The padding and cropping operations taking place in the denoising and deblurring models, respectively, are illustrated in Figure 4.3.

In both cases, the model output is cropped before calculating the loss value, which can be expressed mathematically as multiplication with a sparse *cropping matrix* denoted by $\mathbf{T}$ in [40] and [13]. Inclusion of this matrix changes the loss function from Section 3.1.1 to

$$\begin{aligned}\ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}}) &= -10 \cdot \log_{10}\left(\frac{R^2 \cdot mn}{\|\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}}\|^2}\right) \\ &= -\frac{20}{\ln(10)} \cdot \ln\left(\frac{R \cdot \sqrt{mn}}{\|\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}}\|}\right),\end{aligned}$$

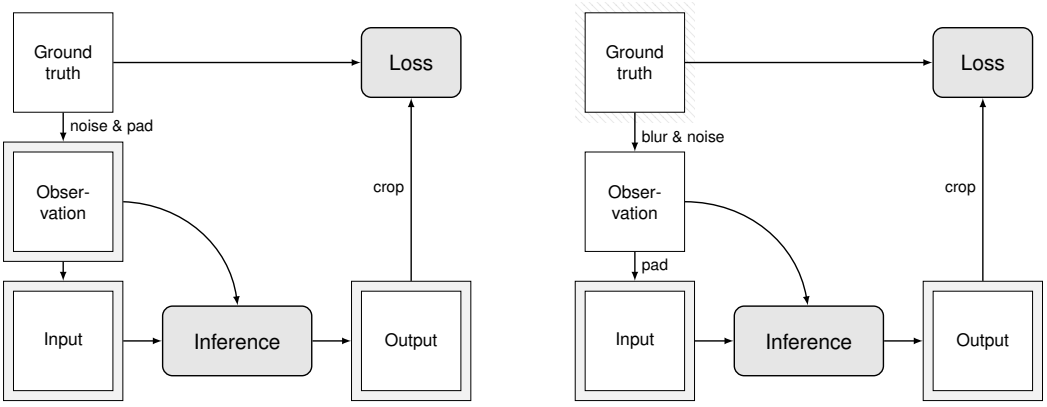


Figure 4.3: Workflow of adding and removing artificial border regions to reduce boundary artifacts for the denoising task *(left)* and for the deblurring task *(right)*. In both cases, an input image is padded with a border that accumulates artifacts and inaccuracies during inference and is removed from the output image before comparison with the ground truth.

assuming the ground truth image $\mathbf{x}_{\mathrm{gt}}$ already has the correct dimensions. The derivative of the loss with respect to the model output $\hat{\mathbf{x}}$ then becomes

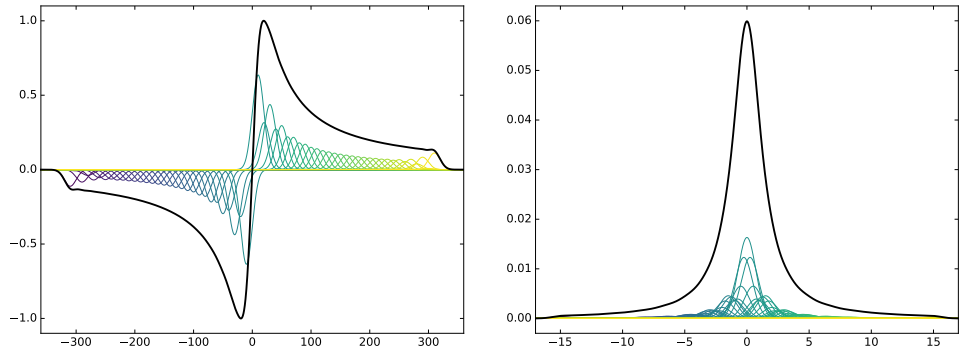
$$\begin{aligned}\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x}_{\mathrm{gt}})}{\partial \hat{\mathbf{x}}} &= \frac{20}{\ln(10)} \cdot \frac{(\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}})^\top}{\|\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}}\|^2} \cdot \mathbf{T} \\ &= \left(\frac{20}{\ln(10)} \cdot \frac{\mathbf{T}^\top(\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}})}{\|\mathbf{T}\hat{\mathbf{x}} - \mathbf{x}_{\mathrm{gt}}\|^2} \cdot \mathbf{T} \right)^\top =: \hat{\mathbf{c}}_T^\top ,\end{aligned}$$

which requires multiplication with the transposed cropping matrix $\mathbf{T}^\top$. As it turns out, multiplication of an image vector with $\mathbf{T}^\top$ is equivalent to padding the image's border with zeros up to the size of an uncropped image. Since this operation as well as the cropping operation can be carried out directly, it is not necessary to construct the matrix $\mathbf{T}$ in practice.

The size of the added border region was chosen to be twice the size of the maximal cliques for the denoising experiments, which amounted to 6 pixels on each side, and half the kernel size, rounded down, for deblurring, which amounted to 13 pixels on each side. While this is not a lot of padding, and larger borders would certainly improve prediction quality, there is a very real tradeoff with respect to computation time, as the complexity of the Cholesky decomposition described in Section 3.4.3 is superlinear in the number of image pixels.

4.1.3 Initialization

Chen and Pock note in [13] that their method, which is essentially the same as the truncated gradient descent model examined here, is not particularly sensitive to initialization when trained greedily. Nonetheless, they propose to initialize the penalty function derivatives modelled by the RBF mixtures ρ_{ti} with a sensible guess. The initial function they choose, which is replicated in the experiments presented



(a) $\rho(u) \approx \frac{2 \cdot (u/20)}{1+(u/20)^2}$ **(b)** $\rho(u) \approx \frac{0.06}{1+0.5 \cdot u^2}$

Figure 4.4: Initial configuration of each radial basis function mixture for one step of **(a)** gradient descent and **(b)** half-quadratic inference. Both initializations are based on the robust *Lorentzian* loss function shown in Figure 2.9b.

here, is the derivative of the Lorentzian loss shown in Figure 2.9b:

$$\rho^*(u) = \alpha \cdot \ln\left(1 + (\beta u)^2\right)$$
$$\rho_{\text{init}}(u) = \frac{\partial \rho^*(u)}{\partial u} = \rho^{*\prime}(u) = \frac{2\alpha\beta^2 \cdot u}{1 + (\beta u)^2}$$

An RBF mixture initialized in this way, with α and β adjusted for the expected range of incoming filter responses, is shown in Figure 4.4a.

Recall from Section 3.4.2 that in the half-quadratic inference model, the RBF mixtures ρ_{ti} model the derivative of the penalty functions divided once more by the argument, that is, the filter response. When the same Lorentzian penalty function as above is chosen for initialization, this yields

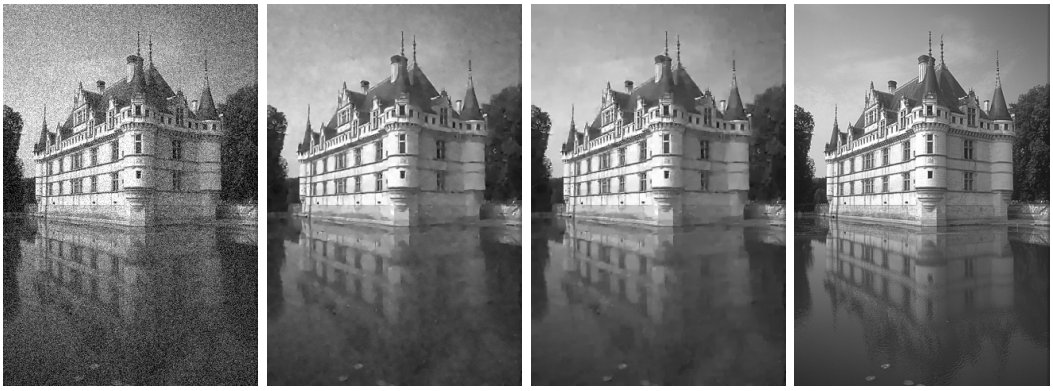
$$\rho_{\text{init}}(u) = \frac{\rho^{*\prime}(u)}{u} = \frac{2\alpha\beta^2}{1 + (\beta u)^2} \,.$$

This function transferred to an RBF mixture is illustrated in Figure 4.4b, again with adjusted α and β. Note that the interval covered by this mixture function is much smaller, reflecting the assumption that for stronger filter responses u the value of $\rho^{*\prime}(u)/u$ can be approximated by zero.

For both the gradient descent and the half-quadratic model, the N filter weights vectors $\tilde{\mathbf{f}}_{ti}$ of every step t are initialized as the columns of the identity matrix $\mathbf{I}^{N \times N}$. This way, the resulting set of initial filters for one step is just the set of filters provided in the filter bank $\mathbf{B}$ as seen in Figure 3.3.

The regularization weight λ_t for the gradient descent model is initially chosen such that $\lambda_t^+ = \ln\left(1 + \exp\left(\lambda_t\right)\right) = 1/10$, a value that is also used in the implementation from [13].

The parameter σ_t in the half-quadratic model is theoretically related to the noise level σ^2 present in the images that need to be restored, and in consequence is initialized to that value. As mentioned in Section 3.4.2, the effect of σ_t could also be achieved by a uniform scaling of the nonlinear functions, so the exact value chosen for it is not of crucial importance.



Observation (20.18 dB) | $\mathrm{GD}^{8}_{3\times3}$ (29.06 dB) | $\mathrm{HQ}^{8}_{3\times3}$ (29.14 dB) | Ground truth

Figure 4.5: Sample output of the two models trained for denoising at noise level $\sigma^2 = 25.0$, compared to observation and ground truth. The PSNR between each image and the ground truth is given in parentheses. The subtle differences may only be visible when zoomed in on a digital screen.

Steps T	1	2	3	4	5	6	7	8
$\mathrm{GD}^{T}_{3\times3}$	27.59	28.04	28.16	28.20	28.22	28.25	28.27	28.28
$\mathrm{HQ}^{T}_{3\times3}$	26.90	27.90	28.14	28.20	28.23	28.24	28.24	28.25
$\mathrm{ARF}^{T}_{5\times5}$ [2]	27.77	28.10	28.17	28.24				
$\mathrm{CSF}^{T}_{3\times3}$ [40]				28.29				
$\mathrm{CSF}^{T}_{5\times5}$ [40]				28.58				
$\mathrm{TNRD}^{T}_{5\times5}$ [13]		28.58			28.78			28.83

Table 4.1: Image denoising results for the gradient descent model $\mathrm{GD}^{T}_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^{T}_{3\times3}$, along with results from similar approaches on the same test set, at noise level $\sigma^2 = 25.0$. All values are in decibels. The half-quadratic model shows convergence after 5 steps, whereas the gradient descent model appears to still improve with additional steps. Overall, gradient descent achieves a better average PSNR on the test set. The results in the lower half should be compared with the caveat that they used more training data and, excepting $\mathrm{CSF}^{T}_{3\times3}$, operated on larger maximal cliques.

4.2 Image Denoising

For the Gaussian denoising task, a gradient descent model $\mathrm{GD}^{T}_{3\times3}$ and a half-quadratic model $\mathrm{HQ}^{T}_{3\times3}$ were trained under the exact same conditions. While the same gradient descent approach was investigated by Chen and Pock in [13], the half-quadratic approach in this form is new. Both trained models have the same capacity, being based on a *Field of Experts* prior with 3×3 cliques and 8 experts, and were trained greedily for $T = 1$ up to $T = 8$ steps. The training set for both models consisted of 50 image patches of size 128×128 and the noise level applied to training and test data was $\sigma^2 = 25.0$. The average PSNR loss over the 68-image denoising test set, evaluated after each step of greedy training, is reported in the upper portion of Table 4.1. The lower portion repeats the results of related work for comparison. Note that these were generally trained on a larger data set, which was not feasible with the half-quadratic model for the reasons stated earlier.

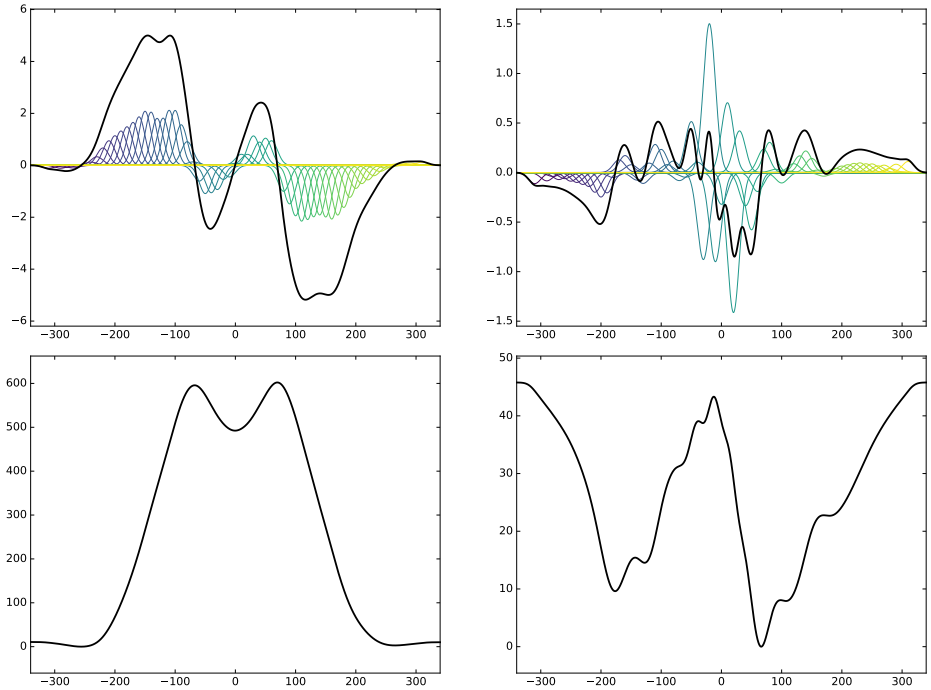


Figure 4.6: Learned nonlinear function $\rho_{t3}(u)$ modelling the response of the third filter in the first step *(upper left)* and last step *(upper right)* of the model $\mathrm{GD}^{8}_{3\times3}$. Pictured in the lower row are the implied penalty functions $\rho^{*}_{t3}(u)$, obtained by approximate integration.

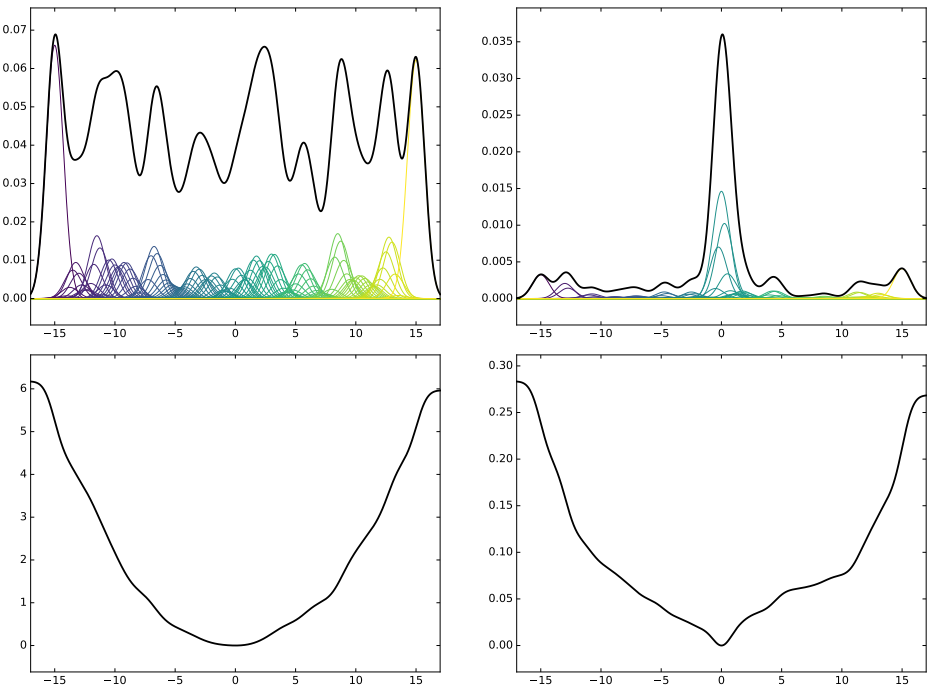


Figure 4.7: Learned nonlinear functions $\rho_{t5}(u)$ modelling the response of the fifth filter in the first step *(upper left)* and last step *(upper right)* of the model $\mathrm{HQ}^{8}_{3\times3}$. Pictured in the lower row are the implied penalty functions $\rho^{*}_{t5}(u)$, obtained by multiplication with u and approximate integration.

Steps T	1	2	3	4	5	6	7	8
$\mathrm{GD}_{3\times3}^{T}$	0.07	0.11	0.15	0.19	0.26	0.30	0.34	0.38
$\mathrm{HQ}_{3\times3}^{T}$	3.79	7.58	11.37	15.15	18.93	22.71	26.50	30.28

Table 4.2: Average test time of the trained denoising models for one image of the test set, in seconds, when stopping after step T. These numbers should be treated with caution, as the code is not well optimized.

4.2.1 Gradient Descent

A sample prediction by the gradient descent model $\mathrm{GD}_{3\times3}^{8}$ for an image from the test set can be seen in Figure 4.5. While the contours of the building and treetops are recovered very clearly, the sky and the lake show a noticeable pattern that has been hallucinated from the noise in the input. A similar effect could be observed in the predictions for several other images with large textureless areas, which are not pictured here. Finer structures of the original image, like the brickwork on the building, are also lost to this pattern. However, these details might be impossible to recover at all when obscured by this level of noise.

The linear filters learned by the model $\mathrm{GD}_{3\times3}^{8}$ can be seen in Figure 6.2. They differ greatly, both between the steps and when compared to the initial filters from Figure 3.3. Two representatives of the nonlinear functions learned by the model are shown in Figure 4.6, along with approximations of the penalty functions they imply. These implied penalty functions correspond to the types (c) and (d) identified by Chen and Pock in [13, Figure 5], suggesting that they have been tuned to sharpen the image and encourage certain textures, respectively. An overview of all penalty functions learned by this model can be found in Figure 6.3, where it can be seen that many diverge from the traditional shape of penalty functions, especially in the later steps of the model.

To investigate this behaviour, which may be connected to the hallucinated textures observed above, the pattern synthesis experiment mentioned in [13, § 4.3] was reproduced with the learned denoising model. To this end, the iteration step defined by a single step of the model was repeated until convergence on an input consisting only of uniform noise. The likelihood part $\lambda_t^{+} \cdot \mathbf{K}^{\top}(\mathbf{K}\hat{\mathbf{x}}_{t-1} - \mathbf{y})$ of the step update formula in Equation (3.33) was dropped here to preclude the influence of the observation $\mathbf{y}$, which is not of interest for this experiment. The results for the second and fifth step of the model $\mathrm{GD}_{3\times3}^{8}$ can be seen in Figure 4.8. Note how the procedure created not only visible, but visibly different textures for these two steps.

While the nonlinear functions learned for the first step display a relatively smooth, at times even piecewise-linear behaviour, the ones belonging to later steps appear much more jagged, as witnessed by the examples in Figure 4.6 and the implied penalty functions shown in Figure 6.3. This may point towards a higher level of specialization in later steps, as their purpose is to correct small errors that are still present in the predictions of their predecessors.

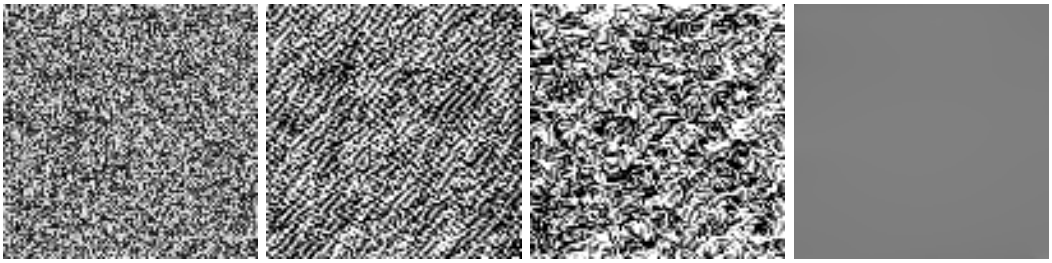


Figure 4.8: Patterns synthesized from uniform noise in the range $[0, 255]$ *(left)* by applying the second step *(center left)* or the fifth step *(center right)* of the trained denoising model $\mathrm{GD}^8_{3\times3}$, without the likelihood term, until convergence. When the same is done using trained step parameters of the denoising model $\mathrm{HQ}^8_{3\times3}$, the image merely converges to a smooth patch of uniform gray *(right)*.

4.2.2 Half-Quadratic Inference

Figure 4.5 also presents the output of the half-quadratic model $\mathrm{HQ}^8_{3\times3}$ for the same test image. The result is very similar to that of the 3×3 gradient descent model, but contains visibly less hallucinated structure in the sky and the lake. This trend is also exhibited in the predictions for other images from the test set. The outputs of the half-quadratic model might therefore appear more pleasing to a human observer, although the overall difference in PSNR values would suggest otherwise. It is possible that a measure more closely modelling human perception, like the *Structural Similarity Index*, would rate these outputs higher. However, the more subtle textures could not be recovered with the half-quadratic approach, either.

The filters learned for each step of the model are presented in Figure 6.4. Notably, the filters learned for all but the first step are essentially unchanged from their initialization, depicted in Figure 3.3. Two nonlinear functions learned by the model, one from the first and one from the last step, are plotted in Figure 4.7. Again, the representative from the last step stayed very close to its initialization, shown in Figure 4.4b. The representative from the first step on the other hand changed significantly. The particularly large weights on the outermost Gaussian RBF components suggest that the function would continue in a similar fashion if was defined on a larger domain. In the lower half of the same figure, reconstructions of the implied penalty functions ρ^*_{ti} can be seen. Both of these bear a strong resemblance to the *truncated quadratic function* and *Tukey's Biweight*, which were shown in Figure 2.9c.

An overview of all implied penalty functions learned by this model can be found in Figure 6.5 of the Appendix. In contrast with the functions learned by the gradient descent model, all of them exhibit the behaviour normally associated with penalty functions, that is, they assign a higher penalty to inputs of higher absolute value. As Chapter 5 will show, this is a consequence of the positivity constraint placed on the learned nonlinear functions. As a result, no generated textures can be observed when conducting the pattern synthesis experiment described above. The steps of the trained half-quadratic denoising model systematically remove noise instead, converging to the uniform gray image seen on the right in Figure 4.8.

Steps T	1	2	3	4	5	6	7	8	30	50
$\mathrm{GD}^T_{3\times3}$	24.08	25.99	27.15	27.99	28.61	29.14	29.56	29.96	33.17	33.92
$\mathrm{HQ}^T_{3\times3}$	32.30	33.42	33.60	33.65	33.69	33.71	33.69	33.71		
$\mathrm{Gauss}_{\mathrm{pw.}}$	32.48									
$\mathrm{GD}^T_{3\times3,\mathrm{init}}$	33.37	33.52	33.67	33.71	33.76	33.81	33.84	33.88		
$\mathrm{HQ}^T_{3\times3,\mathrm{base}}$	33.22	33.76	33.54	33.39	33.29	33.23	33.19	33.16		
$\mathrm{HQ}^T_{\mathrm{pw.,base}}$	32.92	32.44	32.19	32.07	32.00	31.96	31.93	31.91		
$\mathrm{RTF}^T_{5\times5}$ [41]	32.76	33.81								
$\mathrm{CSF}^T_{\mathrm{pw.}}$ [40]	32.48	33.50	33.48							

Table 4.3: Image deblurring results on the deblurring test set from [29] for the gradient descent model $\mathrm{GD}^T_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^T_{3\times3}$ (*first part*), the specially initialized gradient descent model $\mathrm{GD}^T_{3\times3,\mathrm{init}}$ and its Gaussian initialization $\mathrm{Gauss}_{\mathrm{pw.}}$ *(second part)*, two untrained half-quadratic models $\mathrm{HQ}^T_{3\times3,\mathrm{base}}$ and $\mathrm{HQ}^T_{\mathrm{pw.,base}}$ *(third part)* and two related approaches *(fourth part)*. All values are in decibels. When truncated to 8 steps, the gradient descent model performs very poorly here, while the trained half-quadratic model can actually compete with other approaches despite its limited resources. Letting the gradient descent model run for many more steps, or providing it with strong initial guesses, allows it to achieve much better results on this test set. The strong performance of the untrained models compared to the trained ones is discussed in Chapter 5.

4.3 Image Deblurring

For the *non-blind deconvolution* task, the models $\mathrm{GD}^T_{3\times3}$ and $\mathrm{HQ}^T_{3\times3}$ were again trained for $T \in \{1 \dots 8\}$ under indentical conditions, using the same model capacity as in the denoising experiment and a training set of 50 image patches of size 128×128. Both approaches have not been applied to the deblurring problem before. The training data were prepared as detailed in Section 4.1.1 and tests carried out on the test set also described there. The average PSNR achieved by these two models on the 32-image test set is reported in the first portion of Table 4.3, separately for each greedily trained step.

The second part of the same table shows an additional gradient descent experiment, which will be described later on. The third part provides the results of an untrained half-quadratic model with 3×3-cliques, $\mathrm{HQ}^T_{3\times3,\mathrm{base}}$, and an untrained pairwise model $\mathrm{HQ}^T_{\mathrm{pw.,base}}$, on the deblurring test set. In both cases, the filters and nonlinear functions are initialized to those described in Section 4.1.3. These results were intended to represent a baseline for comparison with the trained models. But as it turns out, $\mathrm{HQ}^T_{3\times3,\mathrm{base}}$ outperforms the trained model $\mathrm{HQ}^8_{3\times3}$ in terms of PSNR on the test set, reaching a peak of 33.76. A detailed explanation for this is offered in Chapter 5.

To investigate how much the difference in image and kernel sources between training and test set played into this, the models were evaluated again on an alternative test set. This alternative test set is more similar to the training set, containing 50 images of size 128×128 from the test portion of the *Berkeley Segmentation Data Set*, blurred with another subset of the synthetically generated kernels from [41]. None of these images and blur kernels are contained in the training data. The results for this test

Steps T	1	2	3	4	5	6	7	8	30	50
$\mathrm{GD}^T_{3\times3}$	23.80	25.27	25.99	26.58	26.97	27.34	27.59	27.86	29.97	30.52
$\mathrm{HQ}^T_{3\times3}$	32.27	33.47	33.73	33.81	33.83	33.84	33.81	33.82		
$\mathrm{Gauss}_{\mathrm{pw.}}$	29.32									
$\mathrm{GD}^T_{3\times3,\mathrm{init}}$	31.51	31.56	31.53	31.59	31.56	31.58	31.56	31.61		
$\mathrm{HQ}^T_{3\times3,\mathrm{base}}$	29.69	32.19	33.05	33.35	33.44	33.44	33.40	33.34		
$\mathrm{HQ}^T_{\mathrm{pw.,base}}$	31.28	31.89	31.36	30.94	30.65	30.43	30.27	30.13		

Table 4.4: Image deblurring results for the gradient descent model $\mathrm{GD}^T_{3\times3}$, the half-quadratic model $\mathrm{HQ}^T_{3\times3}$ and the specially initialized gradient descent model $\mathrm{GD}^T_{3\times3,\mathrm{init}}$, evaluated on an alternative test set which is more similar to the training set. In the lower part are the results for the two untrained models. All values are in decibels. On this set, the gradient descent model cannot compete with the trained half-quadratic model even after 50 steps.

set, which are reported in Table 4.4, indicate that the trained half-quadratic model does indeed fare better than the untrained ones on test data that is more similar to its training data. The gradient descent models on the other hand did significantly worse on the alternative test set.

In the last part of Table 4.3, results from two related approaches taken in [41] and [40] are given for comparison. Results for the *Cascade of Regression Tree Fields* $\mathrm{RTF}^T_{5\times5}$ and the *Cascade of Shrinkage Fields* $\mathrm{CSF}^T_{\mathrm{pw.}}$ on the same test set were only available for $T \in \{1, 2\}$ and $T \in \{1, 2, 3\}$, respectively. It should be noted that the same RTF model achieved even better results when training was carried out on a mix of ground truth kernels and estimated kernels, which supports the notion that the unmodified training data insufficiently resemble the test set. Kernel estimation however is not within the scope of this thesis.

4.3.1 Gradient Descent

Training the gradient descent model $\mathrm{GD}^T_{3\times3}$ for $T = 8$ steps produced poor results on both test sets, as reported in Tables 4.3 and 4.4. As a consequence, additional experiments were carried out to see whether there is capacity for improvement.

For the first experiment, the model was trained greedily for an increased number of steps, up to $T = 50$. As the gradient descent approach is many times faster than the half-quadratic one, it was feasible to do this without a reduction of the training set. It turns out that the gradient descent model is able to achieve competitive results on the regular test set when allowed to take many steps to do so. Figure 4.10 suggests that it has a much slower rate of convergence than the analoguous half-quadratic model. In the experiments carried out on the regular deblurring test set, $\mathrm{GD}^T_{3\times3}$ surpassed the half-quadratic model $\mathrm{HQ}^8_{3\times3}$ after 43 iterations, with an average PSNR of 33.73. Whether taking that many steps still qualifies as truncated optimization in the sense of this work may be debatable. However, evaluating all 50 gradient descent steps on an image from the test set is still much faster than carrying out even one step of the half-quadratic model, as shown in Table 4.5.

Steps T	1	2	3	4	5	6	7	8	30	50
$\mathrm{GD}^T_{3\times3}$	0.14	0.28	0.42	0.56	0.70	0.84	0.98	1.12	4.19	6.98
$\mathrm{HQ}^T_{3\times3}$	19.18	38.40	57.57	76.81	96.06	115.38	134.62	153.82		
$\mathrm{GD}^T_{3\times3,\mathrm{init}}$	13.56	13.70	13.84	13.98	14.12	14.26	14.40	14.54		

Table 4.5: Average test time of the trained deblurring models for one image of the test set from [29], in seconds, when stopping after step T. These numbers should be treated with caution, as the code is not well optimized.

All filters and implied penalty functions learned by the model $\mathrm{GD}^8_{3\times3}$ are presented in Figures 6.6 and 6.7. These figures also contain the filters and functions from steps 30 and 50 of the model $\mathrm{GD}^{50}_{3\times3}$. Note that all penalty functions in the first step are of similar shape and have exactly one minimum at zero. Starting from the second step, the majority of penalty functions behave more like the contour and texture encouraging ones described earlier. The trend towards more finely tuned functions in the later steps, observed in the denoising case, is repeated here.

In the second additional experiment, dubbed $\mathrm{GD}^T_{3\times3,\mathrm{init}}$, the basic model architecture again remained the same. But instead of taking the padded observation $\mathbf{y}$ for the initial guess $\hat{\mathbf{x}}_0$, as described in Section 4.1.2, the *maximum a-posteriori* solution of a simple Gaussian model with only pairwise cliques was used (see Section 2.3.4). This preparation step involves the solution of a linear equation system, which is fairly expensive compared to the convolution-based gradient descent steps. Table 4.5 shows it slowing down inference by an average 13.42 seconds for one test set image. Yet, as the results in the second part of Table 4.3 show, the strong initial guess greatly improves the quality of predictions on the regular test set. The average PSNR of the Gaussian initialization $\mathit{Gauss}_{\mathrm{pw.}}$ is also shown, and is in fact higher than the average PSNR after one step of $\mathrm{HQ}^T_{3\times3}$ on this test set.

The filters and implied penalty functions learned for this model can be seen in Figures 6.10 and 6.11. Note how the majority of penalty functions in this model is tuned towards pattern synthesis, starting from the first step. This suggests that the Gaussian initial guess has already removed most of the blur, and the remaining steps primarily aim to sharpen contours and restore details.

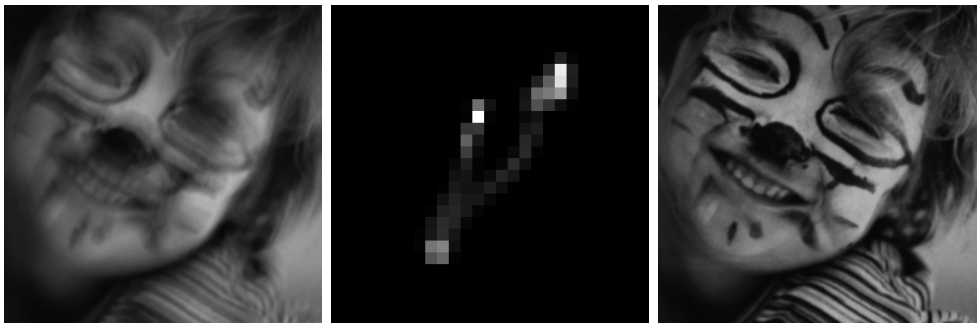
Sample outputs of all three gradient descent models for the same test set image are shown in Figure 4.9, along with the Gaussian initialization and the observation, ground truth and blur kernel. Notably, in this case the gradient descent models produce visibly smoother images, while the output of the half-quadratic model contains a certain level of noise. The same trend is also present in other images from this test set.

4.3.2 Half-Quadratic Inference

In contrast with the gradient descent model, the half-quadratic model $\mathrm{HQ}^8_{3\times3}$ produces good results for the deblurring task with very few steps. As Tables 4.3 and 4.4 and Figure 4.10 show, the average PSNR it achieves on either test set converges around step six. Additional steps could not further improve the quality of its predictions. An example of such a prediction can be seen in Figure 4.9.

The filters and implied penalty functions learned by this model are shown in Figures 6.8 and 6.9. Just as was the case with the half-quadratic denoising model, the filters remained almost unchanged from their initial configuration. The functions however exhibit different behaviours, with none of them taking on the quasi-quadratic form that was prevalent in the denoising model. Instead, many resemble the ℓ_1-norm or the hyper-Laplacian function depicted in Figure 2.9. The first step in particular has multiple penalty functions that supress all negative filter responses and tolerate all positive ones, or vice versa. An example of such a function is shown in Figure 4.12. Note that once again, all functions learned for the half-quadratic model are monotonically increasing for inputs of increasing absolute value, which will be discussed in the next chapter. As a result, the model is not able to perform for the type of pattern synthesis witnessed in the gradient descent models.

The average inference time of $\mathrm{HQ}_{3\times3}^{T}$ for one test image, given in Table 4.5 after each step of the model, is about two orders of magnitude worse than that of the analoguous gradient descent model. Even with a more efficient implementation, it stands to reason that solving the very large equation system based on $\boldsymbol{\Omega}_t$ during each step can not be accomplished with a speed comparable to the convolution-based gradient descent inference. This holds especially for large images, as the equation system matrix grows quadratically in the number of image pixels whereas the computational complexity of convolutions grows in a linear fashion.



Observation (21.22 dB) Blur kernel Ground truth

$\mathrm{GD}^{8}_{3\times3}$ (28.39 dB) $\mathrm{GD}^{50}_{3\times3}$ (33.64 dB) $\mathrm{HQ}^{8}_{3\times3}$ (32.90 dB)

Gaussian init. (32.34 dB) $\mathrm{GD}^{8}_{3\times3,\mathrm{init}}$ (33.88 dB)

Figure 4.9: Predictions by the two basic models $\mathrm{GD}^{8}_{3\times3}$ and $\mathrm{HQ}^{8}_{3\times3}$, as well as the additional models $\mathrm{GD}^{50}_{3\times3}$ and $\mathrm{GD}^{8}_{3\times3,\mathrm{init}}$, all trained for deblurring, for a sample test image blurred with the kernel shown in the top row (not to scale). The solution of a simple Gaussian pairwise model used as input for $\mathrm{GD}^{8}_{3\times3,\mathrm{init}}$ is shown on the bottom left. Observation and ground truth are printed for reference and the PSNR between each image and the ground truth is given in parentheses. The more subtle differences may only be visible when zoomed in on a digital screen.

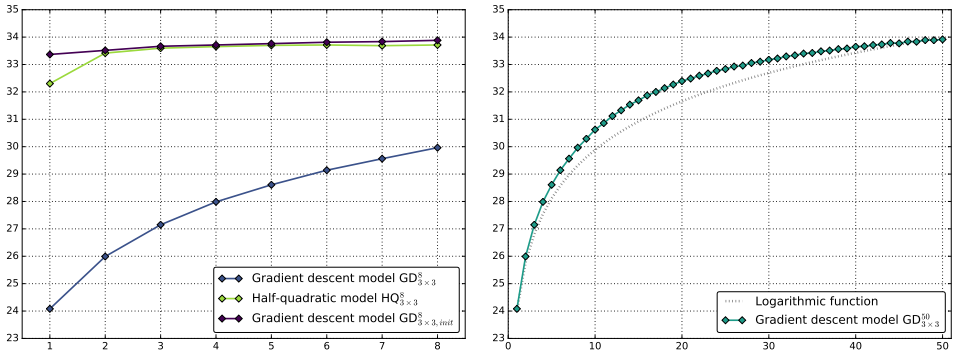


Figure 4.10: Average PSNR value on the deblurring test set after each step of the various deblurring models that were trained. While the half-quadratic model and the quadratically initialized model converge very quickly *(left)*, the pure gradient descent model steadily improves its output over a long number of steps *(right)*.

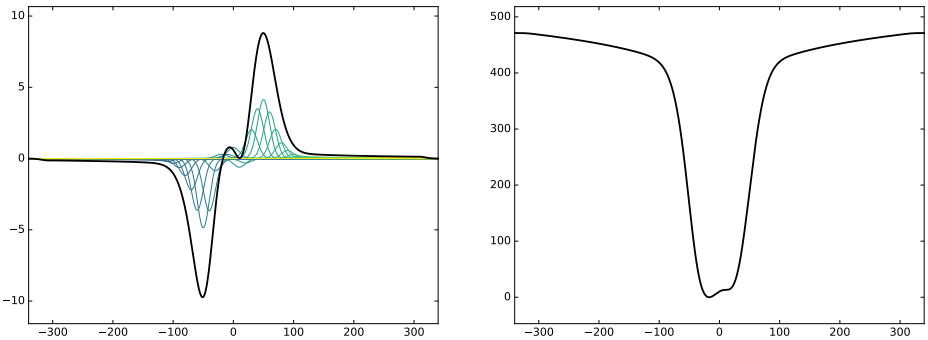


Figure 4.11: Learned nonlinear function $\rho_{1,7}(u)$ modelling the response of the seventh filter in the first step of the model $\mathrm{GD}^8_{3\times3}$ trained for deblurring *(left)*, and the implied penalty function $\rho^*_{1,7}(u)$ obtained by approximate integration *(right)*.

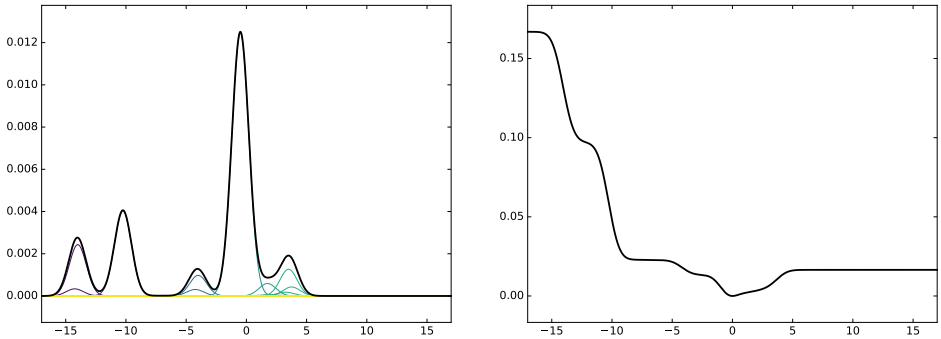


Figure 4.12: Learned nonlinear function $\rho_{1,3}(u)$ modelling the response of the third filter in the first step of the model $\mathrm{HQ}^8_{3\times3}$ trained for deblurring *(left)*, and the implied penalty function $\rho^*_{1,3}(u)$ obtained by multiplication with u and approximate integration *(right)*.

5 Discussion and Outlook

Although the experiments reported in the previous chapter were carried out on a limited training set and with relatively small models, their results nonetheless allow for a number of interesting conclusions and conjectures. These will be presented in the next three sections, followed by an overview of routes that were not investigated here and warrant further research, and finally a short summary of the findings in this thesis.

5.1 Denoising

With the available training data, both the gradient descent and the half-quadratic approach converged to achieve a very similar performance on the test set, with the gradient descent model slightly ahead. The fact that both reached essentially the same ceiling is a hint that more training data and more training iterations may be necessary to find out their full potential. A comparison of the PSNR after eight steps of each model, separately for each test image, is given in Figure 5.1. It mainly demonstrates that none of the two models is consistently better, but that their relative performance also depends on the image content.

Of note is the drastic difference in the filters and implied penalty functions learned by the two models. Where the gradient descent model learned very different filters and functions during each step, the ones learned by the half-quadratic model are much more uniform, and the filters in particular barely differ from their initialization.

As was detailed in the previous chapter, those functions in the gradient descent model which have minima in places other than zero have the effect of sharpening contours and encouraging patterns (see Figure 4.8). The half-quadratic model did not learn any functions which do this, and there is a straightforward reason. It cannot do so because of the positivity constraints placed on its nonlinear functions $\rho(u)$ to ensure that the matrix $\mathbf{\Omega}$ is always *positive-definite*. In Equation (3.48), the nonlinear functions are defined as

$$\rho(u) = \frac{\rho^{*\prime}(u)}{u}\,,$$

which also means that

$$\rho(u) \cdot u = \rho^{*\prime}(u)\,.$$

Because $\rho(u)$ is constrained to be positive, this means that the derivative of the actual penalty function $\rho^*(u)$ is always positive for $u > 0$ and negative for $u < 0$. Therefore, the penalty function is monotonically decreasing for negative inputs and monotonically increasing for positive inputs, leading to a global minimum at zero

and ruling out pattern-generating functions like the ones observed in the gradient descent models.

This restriction of the half-quadratic model leads to smoother output images during denoising, which is visible in the example in Figure 4.5. While the effect is visually pleasing for this particular image, in many cases the noise in the observations is actually obfuscating a non-smooth texture present in the ground truth. Learning to recreate such textures in the right places potentially improves predictions, but it is a quality the constrained half-quadratic model lacks.

Figure 5.2 shows the two test images where the PSNR difference between gradient descent model and half-quadratic model was greatest in favour of the former. Both images contain many finely textured regions and sharp contours, the restoration of which is aided by those sharpening and pattern-generating functions learned by the gradient descent model. The opposite effect can be observed in Figure 5.3, which shows the two test images where the half-quadratic model had the greatest advantage. These are dominated by large homogeneous areas, which are restored especially well by the smoothing effect of the half-quadratic model. This difference serves to highlight the relevance of pattern-generating elements, or lack thereof, in a denoising model.

The fact that the learned half-quadratic model is using practically the same filters it was initialized with may be a hint that more training iterations are needed for it to really adapt to the training data, or that there was a problem with the training algorithm. On the other hand, the similarity of the penalty functions modelling the responses of very different filters, as seen in Figure 6.9, suggests that the exact configuration of each filter may not be as important for this variant of half-quadratic inference. If all filters are modelled by the same kind of function anyway, there might be little value in modifying these filters.

Lastly, the findings in [13] show very clearly that larger maximal cliques lead to better results for the gradient descent denoising model. Whether this also holds for the investigated half-quadratic variant should be determined by additional experiments. However, in light of the very long computation times even when training the 3×3-model, no such experiments were carried out here.

5.2 Deblurring

The first prominent result of the conducted deblurring experiments is that the direct truncated gradient descent approach does not work very well. Within eight steps, it produces worse results on the test set than each of the other models in only a single step. As Figure 5.6 illustrates, it performes worst for the test images blurred with kernels 2 and 4, which are the largest and most widely spread kernels in the test set. The best results of the gradient descent model are obtained on images blurred with kernels 3 and 5, which are much more densely concentrated and cover a smaller area. Since the same tendency can also be observed when comparing the PSNR on each test image individually, visualized in Figure 5.4, the natural conclusion here is that gradient descent-based truncated models do not work well with larger or more complicated blur kernels.

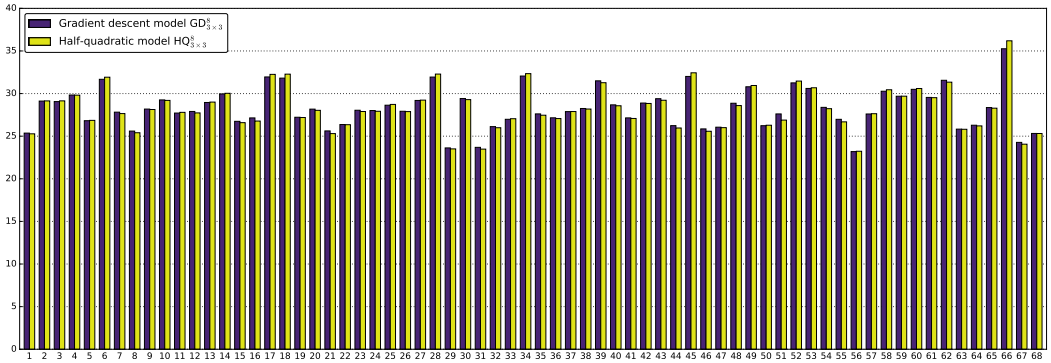


Figure 5.1: PSNR values of the predictions made by the gradient descent model $\mathrm{GD}^8_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^8_{3\times3}$, both trained for denoising, for each image in the denoising test set.

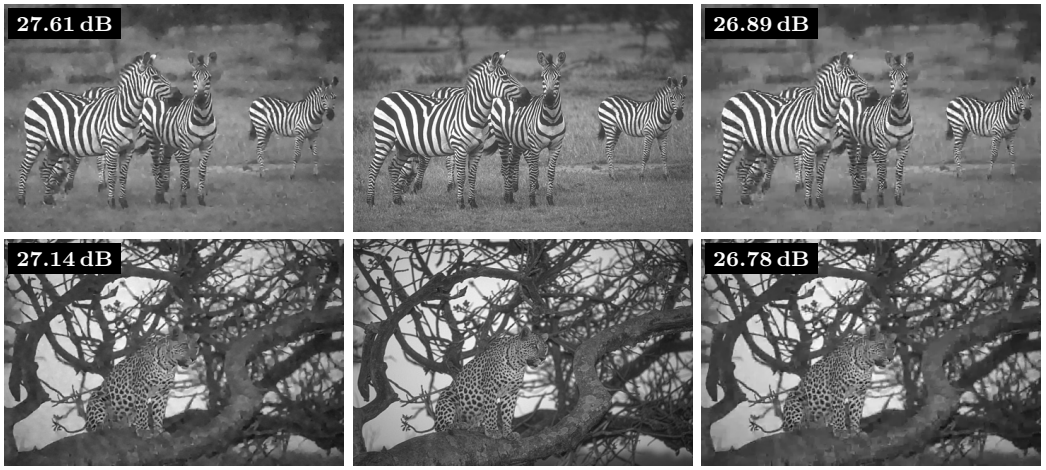


Figure 5.2: Comparison of denoising predictions by the gradient descent model $\mathrm{GD}^8_{3\times3}$ *(left)* and the half-quadratic model $\mathrm{HQ}^8_{3\times3}$ *(right)* for the test images 51 and 16, where the PSNR difference was greatest in favour of the gradient descent model. Ground truths are shown in the middle.

Figure 5.3: Comparison of denoising predictions by the gradient descent model $\mathrm{GD}^8_{3\times3}$ *(left)* and the half-quadratic model $\mathrm{HQ}^8_{3\times3}$ *(right)* for the test images 66 and 18, where the PSNR difference was greatest in favour of the half-quadratic model. Ground truths are shown in the middle.

When truncated only after a sizeable number of steps, the gradient descent approach was shown to produce fairly competitive deblurring results. But as Figures 5.5 and 5.6 show, it still performs worse than the half-quadratic model for the larger kernels 2 and 4, which lends support to the previous point. All in all, for both inference approaches, the deblurring performance seems to depend more on the characteristics of the blur kernel than on the contents of the blurred image. This marks a difference from the denoising problem with constant noise level σ^2, where the corruption in each image is essentially the same, but results vary significantly between different images.

The novel half-quadratic model achieved better results on the test set than the pairwise *Cascade of Shrinkage Fields* model presented in [40], which is based on a different variant of half-quadratic inference, but is not as strong as the larger *Regression Tree Field* model trained in [41]. The model trained here reaches its peak PSNR after 5 to 6 steps, both on the original test set and on the alternative one, indicating that it hit a limit imposed either by the available training data or by the constraints of the model. This convergence can be seen on the left in Figure 4.10. Its computation time, however, constitutes a major drawback, as it took significantly longer per test image than even the half-quadratic denoising model, which operated on larger images. This will be discussed further in the next section.

In contrast with the half-quadratic model, the trained gradient descent model did not seem to have converged even after 50 steps. The plot on the right in Figure 4.10 shows that its progress almost follows a logarithmic trajectory. Since the PSNR value, which is measured in decibels, works on a logarithmic scale, in some sense this means that the output quality improves linearly with each additional gradient descent step. This notion is also supported by Figure 6.1, which shows the time taken by different models to reach a certain PSNR on their respective test sets, on a logarithmic scale. While the other models clearly show convergent behaviour, the deblurring model $\mathrm{GD}^{50}_{3\times3}$ appears to have some potential left. The reason it surpasses the half-quadratic model may again have to do with its more flexible penalty functions. Although its performance on the training set and on the alternative test set is decidedly worse, it has not converged there either and may require even more steps to exploit its full capacity.

Providing the gradient descent model with strong initial guesses for each input image proved very helpful on the regular test set, but less so on the alternative test set. On the former, the initial guesses obtained from a pairwise Gaussian random field are on the same level as 21 steps of ordinary gradient descent (32.49 dB), giving the specially initialized model a good start. It should be noted that from there, the model shows faster convergence than the regular gradient descent model after the 21^{st} step. This suggests that a combination of quadratic or half-quadratic initial guesses followed by gradient descent-based refinement can yield a strong model, exploiting the strong points of both approaches. Table 4.5 shows that the inference time of the combination tested here is dominated by finding the Gaussian solutions. Yet after carrying out 8 gradient descent steps on top of that, it is still faster than one step of the half-quadratic model for deblurring. In practice, a fast and powerful model might be obtained by combining a more efficient method like a *Cascade of Shrinkage Fields* or the *Wiener filter* for the initial guesses with gradient descent refinement. However, care must be taken as these approaches make different assumptions about

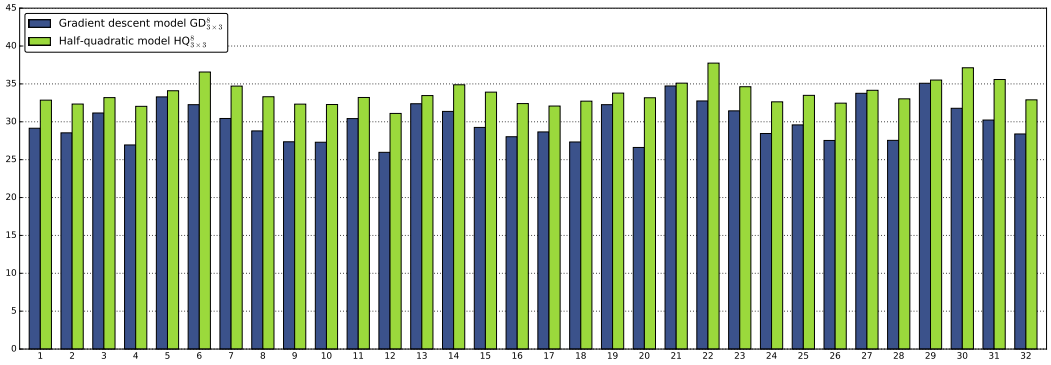


Figure 5.4: PSNR values of the predictions made by the gradient descent model $\mathrm{GD}^{8}_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^{8}_{3\times3}$, both trained for deblurring, for each image in the deblurring test set.

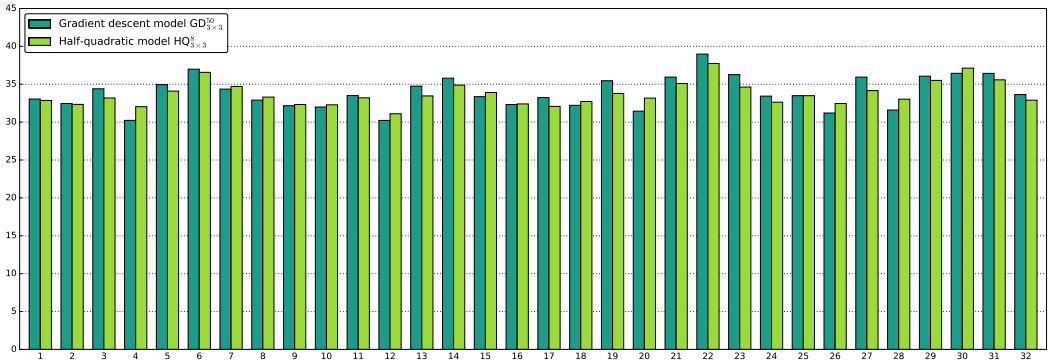


Figure 5.5: PSNR values of the predictions made by the gradient descent model $\mathrm{GD}^{50}_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^{8}_{3\times3}$, both trained for deblurring, for each image in the deblurring test set.

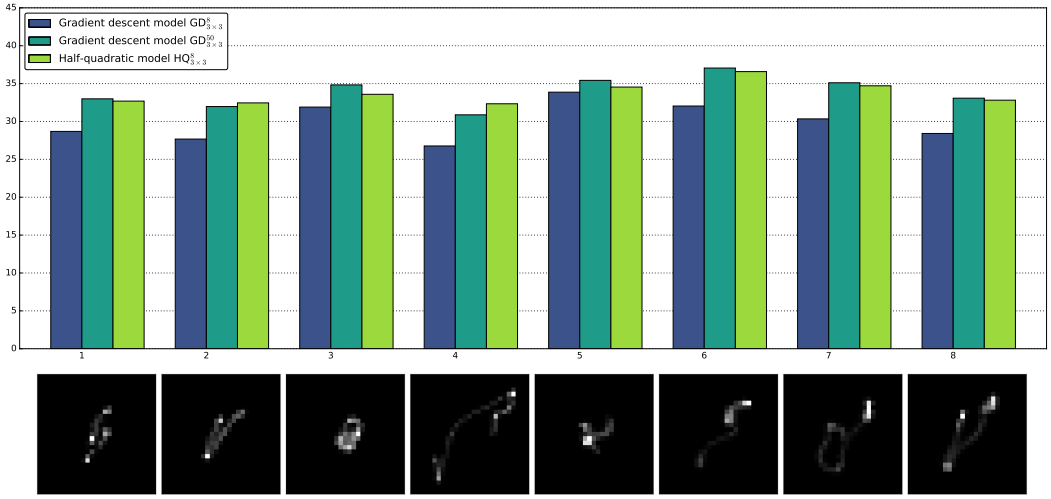


Figure 5.6: Average PSNR values of the predictions made by the gradient descent models $\mathrm{GD}^{8}_{3\times3}$ and $\mathrm{GD}^{50}_{3\times3}$ and the half-quadratic model $\mathrm{HQ}^{8}_{3\times3}$, all trained for deblurring, for the eight different blur kernels in the deblurring test set.

Application	pixels/image	nnz($\mathbf{F}_{ti}$)	nnz($\mathbf{K}$)	nnz($\mathbf{\Omega}_t$)	time/step
Denoising	159 249	1 472 569	159 249	4 079 481	3.6 s
Deblurring	78 961	707 281	7 087 725	28 326 091	19.2 s

Table 5.1: Difference in runtime between half-quadratic denoising and half-quadratic deblurring. The function nnz($\cdot$) counts the number of non-zero elements in a sparse matrix. The large numbers for nnz($\mathbf{K}$) and, as a result, nnz($\mathbf{\Omega}_t$) are the deciding factor for the much longer computation time in the deblurring case. The number of pixels in each padded test image has an influence on nnz($\mathbf{F}_{ti}$), but is less important than the blur kernel size with respect to runtime.

blur effects at the image boundary. When the supplied initial guesses are weak, as is the case in Table 6.1, the combination approach is much less successful.

Finally there is the comparison with the two initialized, but untrained half-quadratic models. The results in Tables 4.3, 4.4 and 6.1 show that the untrained model $\mathrm{HQ}^T_{3\times3,\mathrm{base}}$ outperforms the trained model $\mathrm{HQ}^T_{3\times3}$ on the regular test set, but not on the alternative test set. On the training set, the trained model is significantly stronger. The reason for these very different results may be found in the discrepancies between training and test data for the deblurring experiments. As laid out in Section 4.1.1, the training set uses images from the *Berkeley Segmentation Data Set*, while the images in the test set taken from [29] have different source. In addition, the blur kernels used for training are synthetically generated and may not resemble the kernels in the test set closely enough. For any of these reasons, training that improves a model's performance on the training set may lead to decreased performance on the test set, which is an instance of the *overfitting* problem mentioned in Section 3.1.2. Since the untrained model $\mathrm{HQ}^T_{3\times3,\mathrm{base}}$ is initialized with sensible parameters that are not specialized on any particular data set, it is better able to generalize to the test set.

This notion is supported by the finding in [41] that the RTF approach yields better results on the deblurring test set when its training set contains some ratio of imperfect kernel estimates, helping the model generalize to the slightly different test kernels.

5.3 Computation time

What Tables 4.2 and 4.5 and Figure 6.1 make very clear is the extreme difference in computation time between the two inference approaches. Excepting any further optimization, which will be addressed in the next section, the examined half-quadratic variant is slower than the gradient descent approach by a factor of roughly 100 for denoising and even more for deblurring. To reach a desired PSNR value in either of the applications, the corresponding gradient descent model is consistently much faster than its half-quadratic counterpart, even if it needs to take many more inference steps. In addition, there is a factor of roughly 5 between the inference times of the half-quadratic model for denoising and the one for deblurring.

These large differences have two main reasons. For the difference between gradient descent and half-quadratic inference, it is the dependence of the latter on costly multiplication and decomposition operations on large sparse matrices in each step.

Since the blur matrix $\mathbf{K}$ and the filter matrices $\mathbf{F}_{ti}$ are needed to construct the system matrix $\boldsymbol{\Omega}_t$, they also have to be explicitly constructed, whereas in the gradient descent case, their application can be replaced by much more efficient convolutions.

The difference between deblurring and denoising with the half-quadratic model comes down to the influence of $\mathbf{K}$, which is negligible in the denoising case as $\mathbf{K}$ is simply the identity matrix there. For deblurring, however, the term $\mathbf{K}^\top\mathbf{K}/\sigma^2$ is added to $\boldsymbol{\Omega}_t$ in each step. Because the blur kernels used here are much larger than the linear filters of the models, the sparse blur matrix $\mathbf{K}$ has many more non-zero entries than the filter matrices $\mathbf{F}_{ti}$. As a result, multiplication with its transpose is a lot more expensive. More importantly, its inclusion greatly increases the number of non-zero elements in $\boldsymbol{\Omega}_t$ as well, which leads to much longer computation times for the Cholesky decomposition. Relevant numbers from the experiments are shown in Table 5.1 to illustrate this.

5.4 Outlook

The most important matter for further research on these approaches is improving their runtime, as that is the primary obstacle for more exhaustive and more conclusive experiments. The work of Chen and Pock has already demonstrated that a great speedup is possible for the gradient descent approach, mainly through better use of parallelization and efficient data structures. Although parallelization of the sparse matrix operations used in the half-quadratic approach is tricky and the topic of much research, it would already help to do computations for multiple images in parallel, as these are entirely independent of each other. It is also worth investigating which parts of the half-quadratic approach could be sped up by transferring computations to one or more GPUs, a technique exploited to great effect in [40, 13].

Furthermore, the comparison of the two approaches in this thesis suggests that the potential of the half-quadratic approach is limited by the positivity constraints imposed in Section 3.4.3, as it can not restore texture in the same way the gradient descent model does. It would therefore be interesting to see how the approach fares without these constraints. Relaxing the model in this way certainly precludes the use of the Cholesky decomposition, and it is not immediately clear whether $\boldsymbol{\Omega}_t$ remains at all invertible with entirely unconstrained nonlinear functions. For these reasons, it may be worthwhile to instead investigate the performance of iterative solvers to find $\hat{\mathbf{x}}_t$ from $\boldsymbol{\Omega}_t\hat{\mathbf{x}}_t = \boldsymbol{\eta}_t$. These iterative methods usually depend on a good initialization and a *well-conditioned* system matrix for fast convergence towards an approximate solution [38]. The former can easily be supplied in the form of $\hat{\mathbf{x}}_{t-1}$, since the output of adjacent steps in the model should generally be very similar. However, whether $\boldsymbol{\Omega}_t$ has a favourable *condition number* when the nonlinear functions are unconstrained, or if some kind of *preconditioning* is required for acceptable speed, is not clear and could be the subject of future research.

Once the models are made to run faster, larger maximal cliques may be exploited to increase the model capacity, and larger training sets to ensure good generalization to unseen data. Increased clique sizes have been shown to greatly improve denoising results for the gradient descent model [13], and it can be assumed that they improve

gradient descent deblurring and the half-quadratic approaches for both applications as well.

As to the training procedure, it would be interesting to see what influence jointly training the models has in each case, both with and without previous greedy training. Although the findings in related work show only small improvements with joint training, it is possible that the gradient descent approach to deblurring would benefit from this. Another training related question is the effect of σ^2 in the half-quadratic models, already mentioned in Section 3.4.2. Whether dropping this parameter, or keeping it as some fixed value, would lead to better results could be determined with additional experiments. In addition, the models could be trained and compared again with the *structural similarity index* [43] as a loss measure. The better smoothing effect of the half-quadratic model during denoising, observed in Figure 5.3, might mean that it scores higher under a measure that better resembles human perception.

Lastly, the concept of truncated optimization can be examined for entirely different combinations of model and inference approach, and for related computer vision applications like super-resolution, JPEG deblocking and inpainting. Some of these have already been considered in the related work. Truncation may also work for many different tasks, such as semantic segmentation or stereo reconstruction, to obtain reasonable predictions with a short and fixed inference procedure.

5.5 Conclusion

The findings of Chen and Pock in [13] for the truncated gradient descent approach to denoising could be reproduced here on a smaller scale. It was also demonstrated that the pattern-generating properties, which are a strong point of this model, can still be observed when using only 3×3 cliques.

The novel half-quadratic model investigated in this thesis proved to be on par with the identically trained gradient descent model for the denoising task. Its predictions exhibit a tendency towards smooth surfaces, owing to the constraints placed on the nonlinear functions it learned. As a result, the two trained denoising models were shown to be best suited for different types of image content.

The application of the two approaches to the deblurring task has not been reported before. In the experiments in this thesis, the half-quadratic model consistently produced strong results on different data sets, reaching its highest score after 6 steps in each case. The experiments confirmed the expectation that a similar gradient descent model with the same number of steps is not well suited for deblurring. When truncating after a much larger number of steps, however, it proved able to outperform the compared models on the test set taken from [29]. Unfortunately, this positive finding is challenged by the much poorer results of the same model on a second test set. While no conclusive reason could be found for this disparity, it may be connected to the different characteristics of the blur kernels in each set, as the quality of the gradient descent predictions seems to depend on kernel size and spread.

The deblurring results of the gradient descent model were also improved by supplying it with good initial guesses from a different method. The model might therefore prove useful as a post-processing step, refining the output of other deblurring systems.

The crucial problem with the otherwise convincing half-quadratic model is its runtime, which is two orders of magnitude greater than that of an analoguous gradient descent model. This was to be expected to some extent, since the approach depends on expensive operations on large matrices in each iteration. Yet unless there is a way to speed these calculations up dramatically, the half-quadratic models in this thesis can not deliver on the fast runtime that is one of the core motivations for truncated inference in general.

A direct comparison with the results obtained by related approaches was not possible, because the models trained here operate on smaller cliques and smaller training sets than those previously published. An interesting aside, however, is how well the untrained half-quadratic model performed on the deblurring test set from [29], surpassing the CSF model [40] and getting close to the results of the RTF model [41]. This could mean that the considered half-quadratic variant is especially well suited for the deblurring task. It could however also mean that there is a flaw with the test set, which might also explain the immense performance difference for the gradient descent model in comparison with an alternative test set.

To improve the results of both models, their code should be made more efficient to allow for larger cliques and data sets. In addition, a modification of the half-quadratic model relaxing the constraints on its nonlinear functions could improve its denoising of textured regions. For deblurring, the most obvious improvement may be found in obtaining more efficient strong initializations for the gradient descent model.

6 Appendix

This chapter provides space for a selection of plots and data which would have taken up too much room in the main part of the thesis. Context for these figures and a discussion of their content can be found in Chapters 4 and 5.

Steps T	1	2	3	4	5	6	7	8	30	50
$\mathrm{GD}^T_{3\times3}$	22.70	24.15	24.87	25.40	25.79	26.15	26.40	26.65	28.43	28.85
$\mathrm{HQ}^T_{3\times3}$	31.55	32.83	33.15	33.28	33.32	33.32	33.31	33.32		
$\mathrm{Gauss}_{\mathrm{pw.}}$	27.29									
$\mathrm{GD}^T_{3\times3,\mathrm{init}}$	28.21	28.44	28.55	28.61	28.66	28.70	28.74	28.76		
$\mathrm{HQ}^T_{3\times3,\mathrm{base}}$	28.56	31.26	32.19	32.56	32.71	32.76	32.75	32.72		
$\mathrm{HQ}^T_{\mathrm{pw.,base}}$	30.02	31.06	30.57	30.19	29.92	29.72	29.55	29.41		

Table 6.1: Image deblurring results for the gradient descent model $\mathrm{GD}^T_{3\times3}$, the half-quadratic model $\mathrm{HQ}^T_{3\times3}$ and the specially initialized gradient descent model $\mathrm{GD}^T_{3\times3,\mathrm{init}}$, evaluated on the data set they were trained with. The lower part shows the results for the two untrained half-quadratic models on the same set.

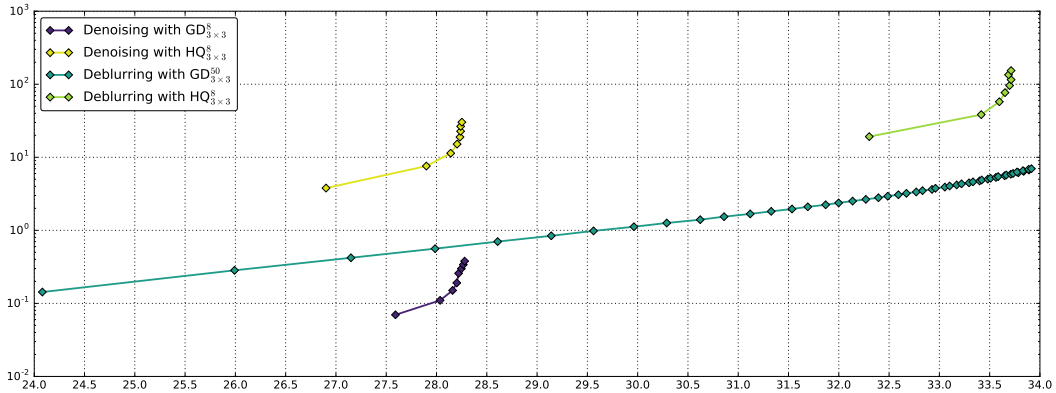


Figure 6.1: Time taken by the different trained models to achieve a given average PSNR on their respective test set. Time in seconds is plotted on a logarithmic scale to accomodate the extreme differences between gradient descent and half-quadratic models. Note how the progression for $\mathrm{GD}^{50}_{3\times3}$ is almost linear in this representation, suggesting that it is not fully converged yet.

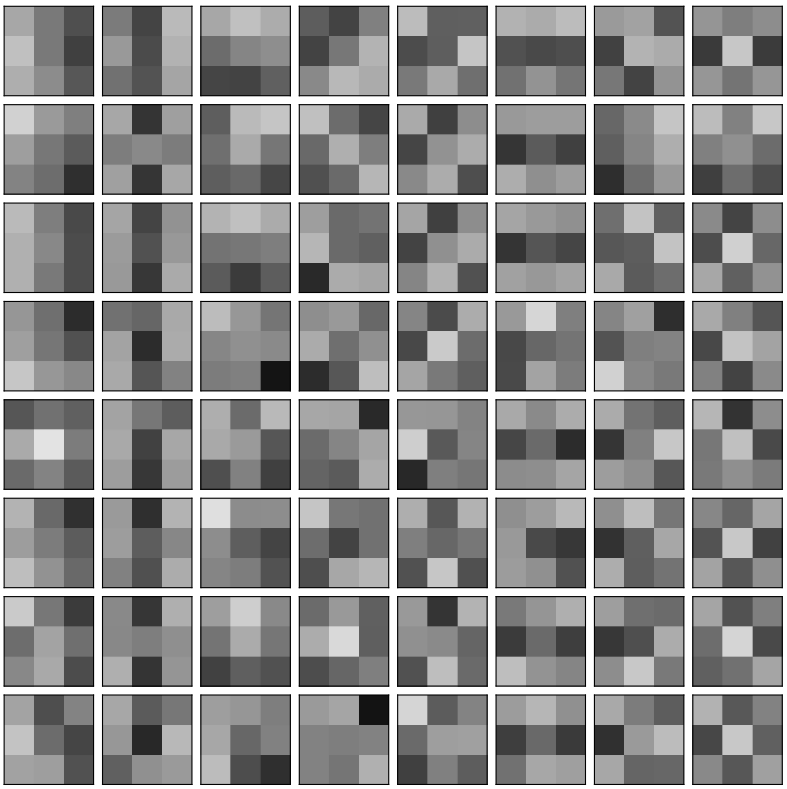


Figure 6.2: All filters learned for the gradient descent denoising model $\mathrm{GD}^8_{3\times 3}$. Each row represents one step and each column corresponds to one expert.

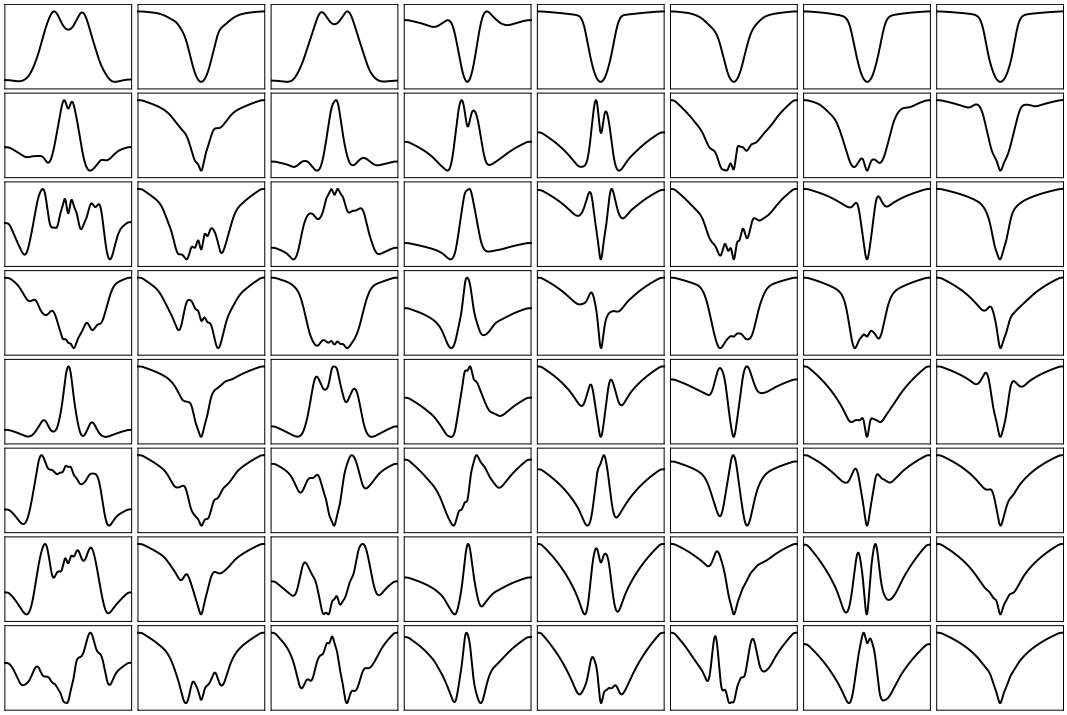


Figure 6.3: All penalty functions learned by the model $\mathrm{GD}^8_{3\times 3}$ for denoising, as obtained via approximate integration. Each row represents one step and each column corresponds to one expert.

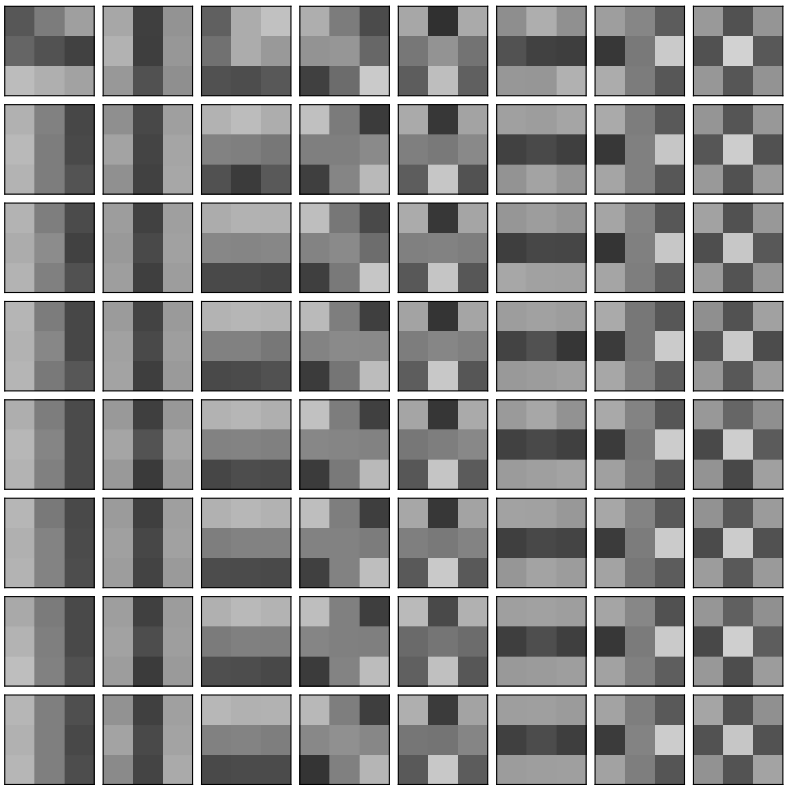


Figure 6.4: All filters learned for the half-quadratic denoising model $\mathrm{HQ}^{8}_{3\times3}$. Each row represents one step and each column corresponds to one expert.

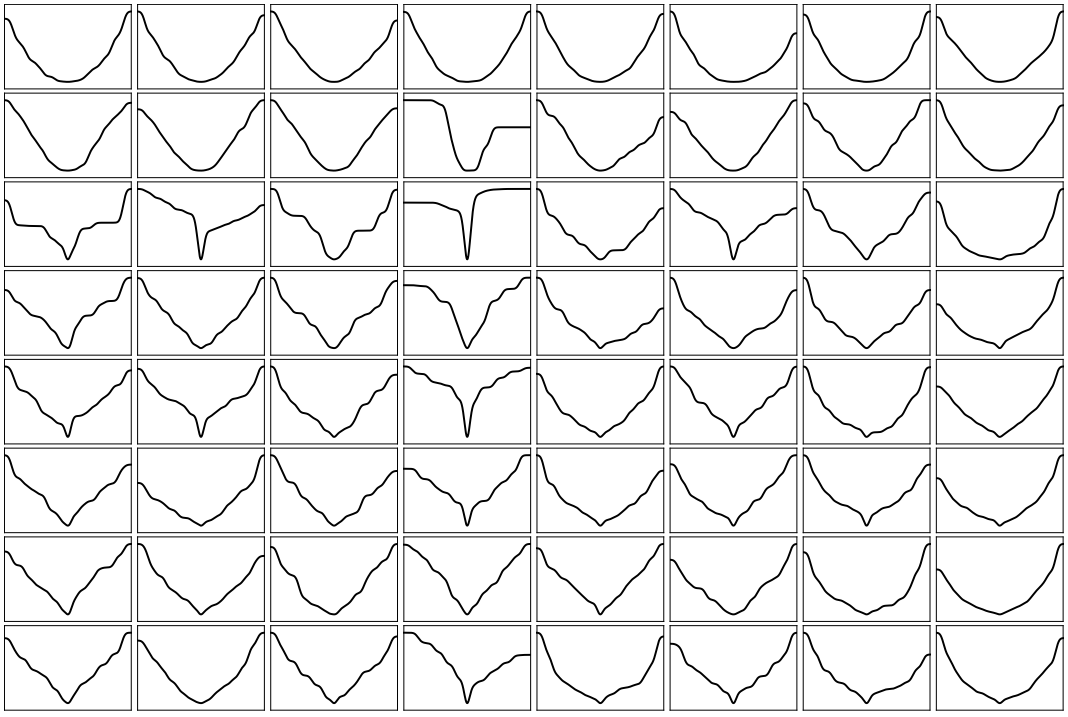


Figure 6.5: All penalty functions learned by the model $\mathrm{HQ}^{8}_{3\times3}$ for denoising, as obtained via approximate integration. Each row represents one step and each column corresponds to one expert.

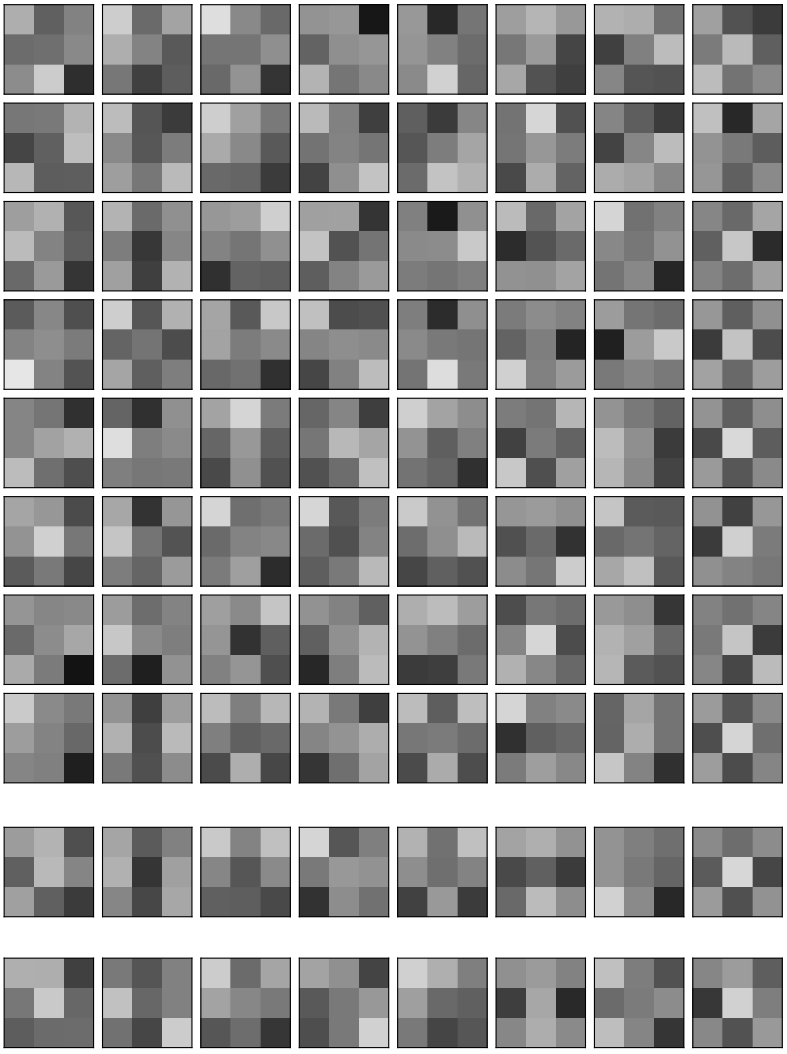


Figure 6.6: All filters learned for the gradient descent deblurring model $\mathrm{GD}^{8}_{3\times3}$, and the ones learned by the model $\mathrm{GD}^{50}_{3\times3}$ for step 30 and 50. Each row represents one step and each column corresponds to one expert.

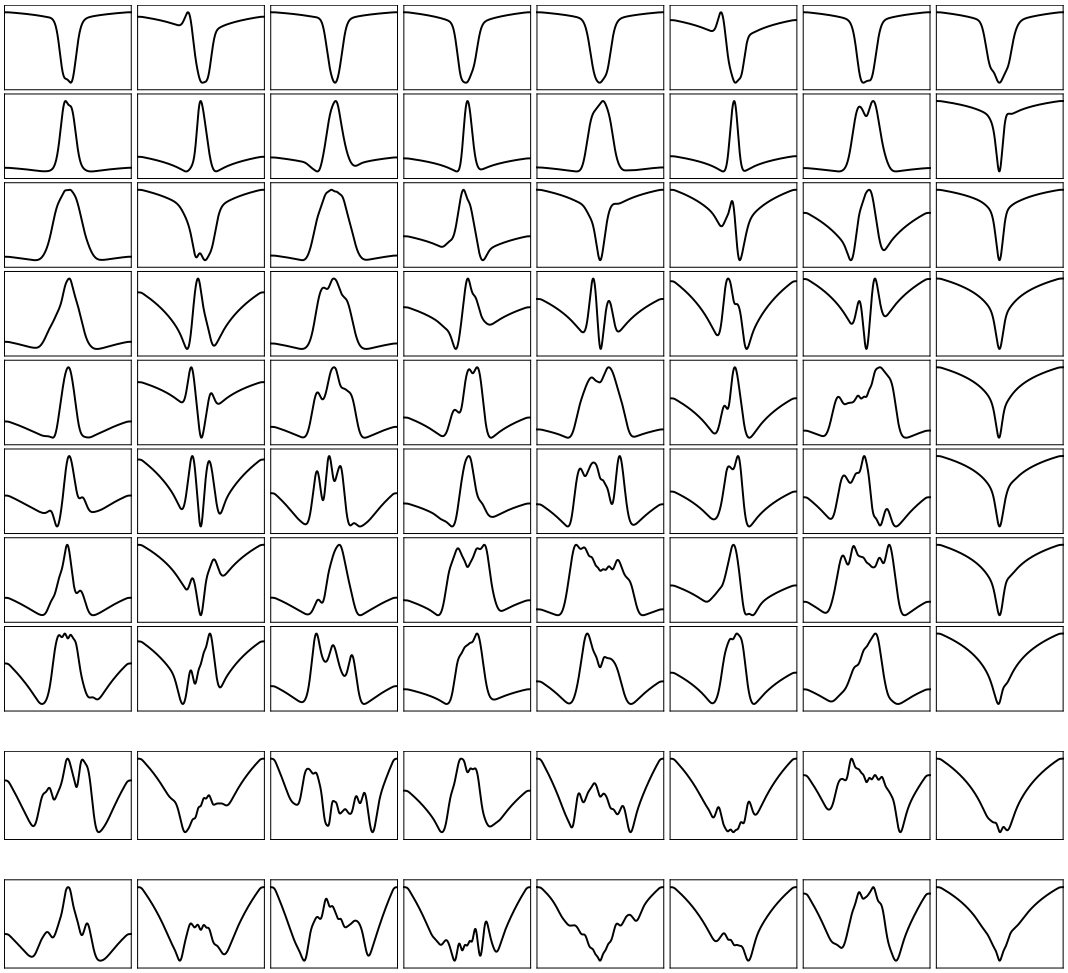


Figure 6.7: All penalty functions learned by the model $\mathrm{GD}^{8}_{3\times 3}$ for deblurring, and the ones learned by the model $\mathrm{GD}^{50}_{3\times 3}$ for step 30 and 50, all obtained via approximate integration. Each row represents one step and each column corresponds to one expert.

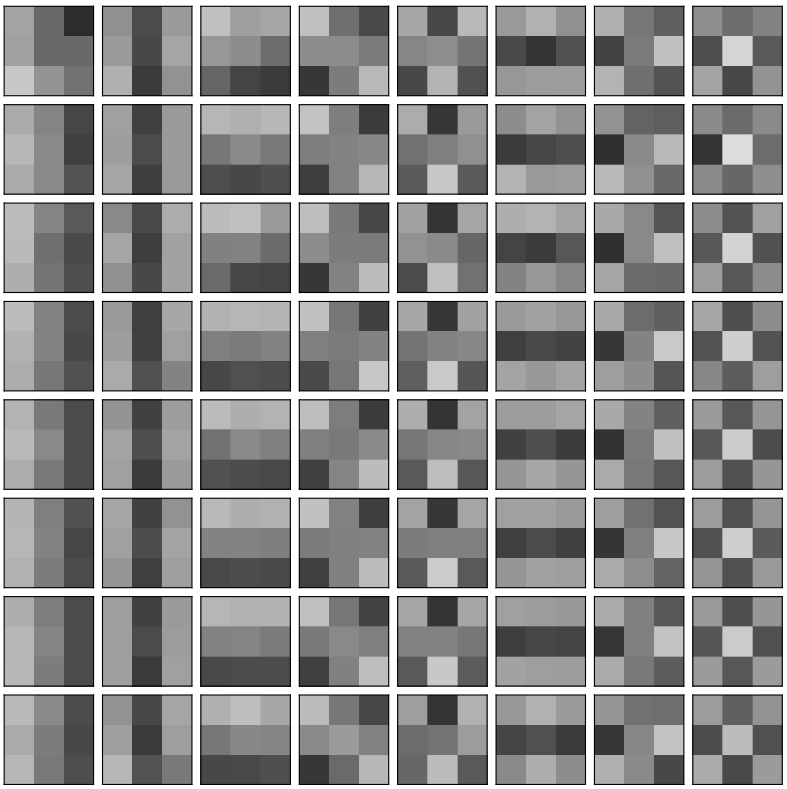


Figure 6.8: All filters learned for the half-quadratic deblurring model $\mathrm{HQ}^8_{3\times3}$. Each row represents one step and each column corresponds to one expert.

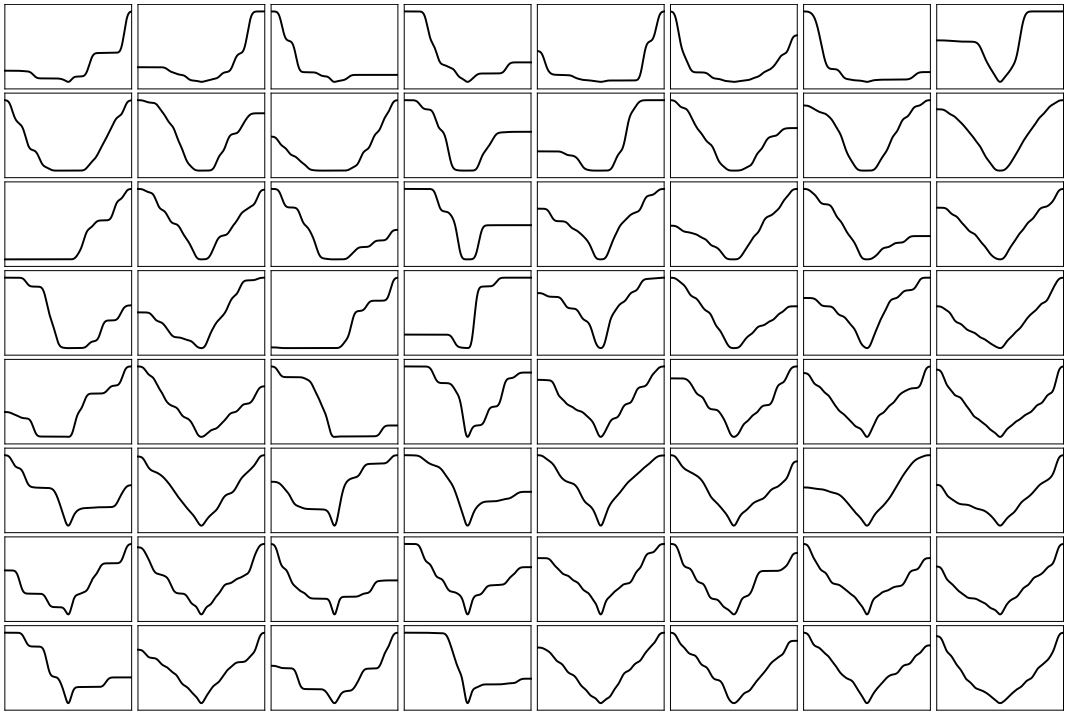


Figure 6.9: All penalty functions learned by the model $\mathrm{HQ}^8_{3\times3}$ for deblurring, as obtained via approximate integration. Each row represents one step and each column corresponds to one expert.

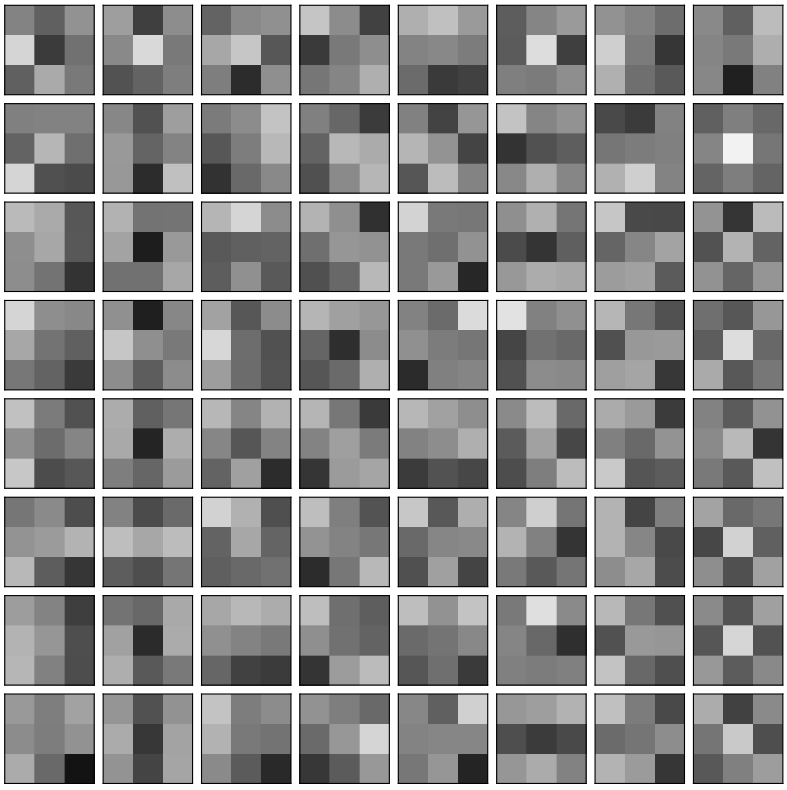


Figure 6.10: All filters learned for the specially initialized gradient descent deblurring model $\mathrm{GD}^{8}_{3\times3,\mathrm{init}}$. Each row represents one step and each column corresponds to one expert.

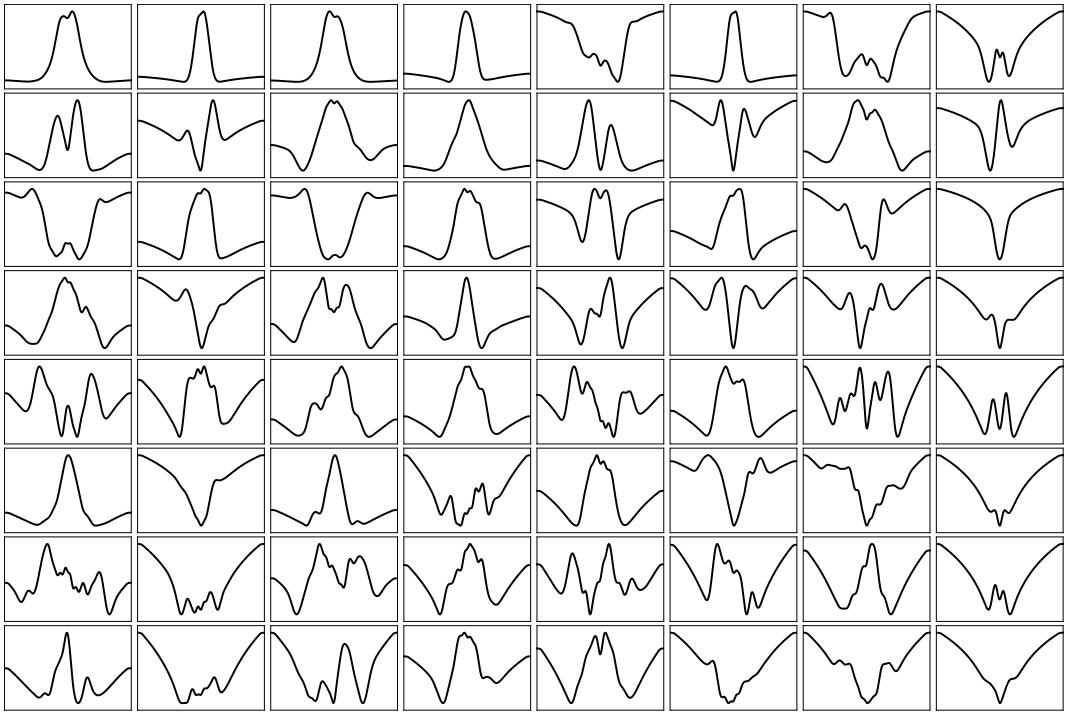


Figure 6.11: All penalty functions learned by the model $\mathrm{GD}^{8}_{3\times3,\mathrm{init}}$ for deblurring, as obtained via approximate integration. Each row represents one step and each column corresponds to one expert.

Bibliography

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5 (May 2011), pp. 898–916. DOI: 10.1109/TPAMI.2010.161.

[2] Adrian Barbu. "Training an Active Random Field for Real-Time Image Denoising". In: *IEEE Transactions on Image Processing* 18.11 (Nov. 2009), pp. 2451–2462. DOI: 10.1109/TIP.2009.2028254.

[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. *Julia: A Fresh Approach to Numerical Computing*. Nov. 2014. arXiv: 1411.1607 [cs.MS].

[4] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. *Julia: A Fast Dynamic Language for Technical Computing*. Sept. 2012. arXiv: 1209.5145 [cs.PL].

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007.

[6] Michael J. Black and Anand Rangarajan. "On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision". In: *International Journal of Computer Vision* 19.1 (July 1996), pp. 57–91. DOI: 10.1007/BF00131148.

[7] Mikael Boden. "A Guide to Recurrent Neural Networks and Backpropagation". In: *The Dallas Project, SICS Technical Report* (2002).

[8] Ronald Bracewell. *The Fourier Transform & Its Applications*. English. 3rd Edition. New York: McGraw-Hill, June 1999, pp. 108–112.

[9] Charles G. Broyden. "Quasi-Newton Methods and Their Application to Function Minimisation". In: *Mathematics of Computation* 21.99 (1967), pp. 368–381. DOI: 10.1090/S0025-5718-1967-0224273-2.

[10] Augustin-Louis Cauchy. "Méthode générale pour la résolution des systèmes d'équations simultanées". In: *Compte Rendu des S'eances de L'Acad'emie des Sciences XXV* S'erie A.25 (Oct. 1847), pp. 536–538.

[11] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. "Two Deterministic Half-Quadratic Regularization Algorithms for Computed Imaging". In: *IEEE International Conference on Image Processing*. Vol. 2. Nov. 1994, pp. 168–172. DOI: 10.1109/ICIP.1994.413553.

[12] Priyam Chatterjee, Neel Joshi, Sing Bing Kang, and Yasuyuki Matsushita. "Noise Suppression in Low-light Images Through Joint Denoising and Demosaicing". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 321–328. DOI: 10.1109/CVPR.2011.5995371.

[13] Yunjin Chen and Thomas Pock. *Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration*. Aug. 2015. arXiv: 1508.02848 [cs.CV].

[14] Yunjin Chen, Thomas Pock, René Ranftl, and Horst Bischof. "Revisiting Loss-Specific Training of Filter-Based MRFs for Image Restoration". In: *Pattern Recognition*. Ed. by Joachim Weickert, Matthias Hein, and Bernt Schiele. Lecture Notes in Computer Science 8142. Springer Berlin Heidelberg, Sept. 2013, pp. 271–281. DOI: 10.1007/978-3-642-40602-7.

[15] Michael A. Covington. *Digital SLR Astrophotography*. Cambridge University Press, Nov. 2007.

[16] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006.

[17] Justin Domke. "Generic Methods for Optimization-Based Modeling". In: *International Conference on Artificial Intelligence and Statistics*. Ed. by Neil D. Lawrence and Mark A. Girolami. Vol. 22. 2012, pp. 318–326.

[18] Mário A. T. Figueiredo. "On Gaussian Radial Basis Function Approximations: Interpretation, Extensions, and Learning Strategies". In: *15th International Conference on Pattern Recognition*. Vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, 2000, pp. 618–621. DOI: 10.1109/ICPR.2000.906151.

[19] Donald Geman and George Reynolds. "Constrained Restoration and the Recovery of Discontinuities". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.3 (Mar. 1992), pp. 367–383. DOI: 10.1109/34.120331.

[20] Donald Geman and Chengda Yang. "Nonlinear Image Recovery with Half-quadratic Regularization". In: *IEEE Transactions on Imgage Processing* 4.7 (July 1995), pp. 932–946. DOI: 10.1109/83.392335.

[21] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *Readings in Computer Vision*. San Francisco (CA): Morgan Kaufmann, 1987, pp. 562–563.

[22] Christine Guillemot and Olivier Le Meur. "Image Inpainting: Overview and Recent Advances". In: *IEEE Signal Processing Magazine* 31.1 (2014), pp. 127–144.

[23] J. M. Hammersley and P. Clifford. *Markov Fields on Finite Graphs and Lattices*. Unpublished manuscript, 1971.

[24] Yu Hen Hu and Jenq-Neng Hwang. *Handbook of Neural Network Signal Processing*. Electrical Engineering & Applied Signal Processing. CRC Press, 2001.

[25] Jinggang Huang and David Mumford. "Statistics of Natural Images and Models". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE Computer Society, 1999, p. 547. DOI: 10.1109/CVPR.1999.786990.

[26] Jeremy Jancsary, Sebastian Nowozin, and Carsten Rother. "Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art". In: *European Conference on Computer Vision*. Springer, Aug. 2012.

[27] Ross Kindermann and J. Laurie Snell. *Markov Random Fields and Their Applications*. Vol. 1. Contemporary Mathematics. Providence, Rhode Island: American Mathematical Society, 1980.

[28] Dilip Krishnan and Rob Fergus. "Fast Image Deconvolution Using hyper-Laplacian Priors". In: *International Conference on Neural Information Processing Systems*. Vancouver, British Columbia, Canada: Curran Associates Inc., 2009, pp. 1033–1041.

[29] Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman. "Understanding and Evaluating Blind Deconvolution Algorithms". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 1964–1971.

[30] Xin Li, Bahadir Gunturk, and Lei Zhang. "Image Demosaicing: A Systematic Survey". In: *SPIE Conference on Visual Communications and Image Processing*. Vol. 6822. Jan. 2008, DOI: 10.1117/12.766768.

[31] D. C. Liu and J. Nocedal. "On the Limited Memory BFGS Method for Large Scale Optimization". In: *Mathematical Programming* 45.3 (Dec. 1989), pp. 503–528. DOI: 10.1007/BF01589116.

[32] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*. Tech. rep. 2006.

[33] Michael J. D. Powell. "Radial Basis Functions for Multivariable Interpolation: A Review". In: *Algorithms for Approximation*. New York, NY, USA: Clarendon Press, 1987, pp. 143–167.

[34] Stefan Roth and Michael J. Black. "Fields of Experts: A Framework for Learning Image Priors". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE Computer Society, June 2005, pp. 860–867. DOI: 10.1109/CVPR.2005.160.

[35] Stefan Roth and Michael J. Black. "Fields of Experts". In: *International Journal of Computer Vision* 82.2 (Apr. 2009), pp. 205–229. DOI: 10.1007/s11263-008-0197-6.

[36] Håvard Rue and Leonhard Held. *Gaussian Markov Random Fields: Theory and Applications*. Vol. 104. Monographs on Statistics and Applied Probability. London: Chapman & Hall, 2005.

[37] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*. MIT Press, 1986. Chap. 8, pp. 318–362.

[38] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 2nd. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[39] Kegan G. G. Samuel and Marshall F. Tappen. "Learning Optimized MAP Estimates in Continuously-Valued MRF Models". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, June 2009, pp. 477–484. DOI: 10.1109/CVPR.2009.5206774.

[40] Uwe Schmidt and Stefan Roth. "Shrinkage Fields for Effective Image Restoration". In: *IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 2774–2781.

[41] Uwe Schmidt, Carsten Rother, Sebastian Nowozin, Jeremy Jancsary, and Stefan Roth. "Discriminative Non-blind Deblurring". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Apr. 2013.

[42] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.

[43] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[44] Jianxin Wu. *Some Properties of the Gaussian Distribution*. Georgia Institute of Technology, 2004.

[45] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. "Single-Image Super-Resolution: A Benchmark". In: *European Conference on Computer Vision*. Springer. 2014, pp. 372–386.

Erklärung

Hiermit erkläre ich, dass ich die am 12. 09. 2016 eingereichte Masterarbeit zum Thema *Comparison of Learned Inference Approaches for Image Restoration* unter Betreuung von Prof. Carsten Rother selbstständig erarbeitet, verfasst und Zitate kenntlich gemacht habe. Andere als die angegebenen Hilfsmittel wurden von mir nicht benutzt.

Dresden, den 12. 09. 2016 Jakob Kruse